

MÓDULO 1

Conceitos Introdutórios

RESUMO

O mundo vive a era digital: na atualidade os circuitos e técnicas digitais passaram a ser amplamente utilizados em quase todas as áreas: computadores, automação, robôs, tecnologia e ciência médica, transportes, entretenimento, exploração espacial, etc. Nesse sentido, conhecer os princípios, os conceitos e as operações fundamentais que são comuns aos sistemas digitais, desde uma simples chave liga/desliga até o mais complexo computador é muito importante quando da análise e manutenção de qualquer sistema digital.

OBJETIVO

O objetivo desse módulo é a introdução de alguns conceitos fundamentais que são imprescindíveis no estudo e entendimento da tecnologia digital, bem como o conhecimento das terminologias utilizadas no mundo digital.

CONTEÚDO

1. Quantidades analógicas X Quantidades Digitais

As quantidades analógicas podem assumir quaisquer valores ao longo de uma faixa contínua. Exemplos disso são a corrente que flui de uma tomada elétrica, a temperatura de um ambiente, o velocímetro de um carro, etc.

Quanto às quantidades digitais podem assumir apenas valores discretos. Como exemplo pode-se citar os grãos de areia em uma praia, uma vez que estes são valores discretos (inteiros), não sendo possível ter valores ao longo de uma faixa contínua. Outro exemplo bastante familiar é o relógio digital, o qual apresenta a hora do dia na forma de dígitos decimais, que representam as horas, os minutos e os segundos. Apesar de o tempo variar continuamente, o relógio digital varia em saltos ou degraus.

2. Sistemas Analógicos X Quantidades Digitais

Um *sistema digital* é a combinação de dispositivos projetados para manipular informação lógica ou quantidades físicas que são representadas no formato digital. Esses dispositivos podem ser eletrônicos, mecânicos, magnéticos e pneumáticos. Exemplos de sistemas digitais são os computadores e as calculadoras, os equipamentos digitais de áudio e vídeo, o sistema de telefonia, etc.

Um *sistema analógico* contém dispositivos que manipulam quantidades físicas que são representadas na forma analógica. Exemplos: amplificadores de áudio, a amplitude do sinal de saída de um alto-falante em um receptor de rádio, etc.

2.1. Vantagens e Limitações das Técnicas Digitais

As vantagens das técnicas digitais sobre as analógicas são: fáceis de projetar, fáceis de armazenar informações, maior precisão e exatidão, programabilidade, menos afetadas por ruído, maior grau de integração.

A principal limitação no uso das técnicas digitais é que no mundo real, a maioria das quantidades físicas é de natureza analógica, que são muitas vezes as entradas e saídas monitoradas, operadas e controladas por um sistema, fazendo-se necessário, portanto, convertê-las para o formato digital para serem processadas, e voltá-las ao formato analógico. A figura 1 mostra um diagrama com esses passos, de um sistema de controle de temperatura.

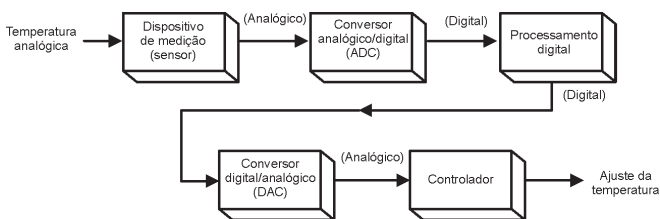


Figura 1 - Diagrama de um sistema de controle de temperatura que requer conversão analógico-digital para permitir o uso de técnicas de processamento digital - (Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer)

3. Sistema Binário

É o sistema básico de numeração usado em quase todos os sistemas digitais

3.1. Numeração Binária

Utiliza somente os algarismos **0** e **1** como códigos. Representa números decimais e letras do alfabeto.

- **Definições importantes:**

Bit – 1 dígito binário

Nibble – 4 dígitos binários

Byte – 8 dígitos binários

MSB – bit mais significativo (most significant bit – MSB)

LSB – bit menos significativo (least significant bit – MSB)

3.2. Seqüência de Contagem Binária

A figura 2 ilustra a seqüência de contagem binária.

Pesos →	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$		Número decimal equivalente
	0	0	0	0	→	0
	0	0	0	1	→	1
	0	0	1	0		2
	0	0	1	1		3
	0	1	0	0		4
	0	1	0	1		5
	0	1	1	0		6
	0	1	1	1		7
	1	0	0	0		8
	1	0	0	1		9
	1	0	1	0		10
	1	0	1	1		11
	1	1	0	0		12
	1	1	0	1		13
	1	1	1	0	→	14
	1	1	1	1	→	15
				↑		
				LSB		

Figura 2 - Seqüência de contagem binária - (Sistemas Digitais:

3.4. Representação de Quantidades Binárias

As quantidades binárias podem ser representadas por qualquer dispositivo que tenha apenas dois estados de operação ou duas condições possíveis, como por exemplo, uma chave que pode estar aberta (binário 0) ou fechada (binário 1). Na figura 3 a seguir, os estados das diversas chaves representam 10010_2 .

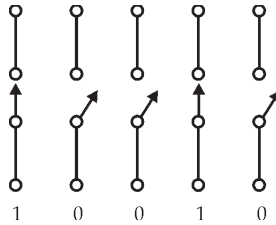


Figura 3 - Chaves representando 0 (aberta) e 1 (fechada) - (Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer)

4. Sinais Digitais e Diagramas de Tempo

As informações binárias em um sistema digital são representadas por tensões ou correntes presentes nos circuitos. Tipicamente, tem-se:

- Zero volt (0 V) representa o binário 0;
- + 5 V representa o binário 1.

Devido às variações nos circuitos, o 0 e o 1 são representados por faixas de tensão, conforme ilustra a figura 4(a).

Um gráfico que mostra como um ou mais sinais digitais variam ao longo do tempo é denominado diagrama de tempo, figura 4(b).

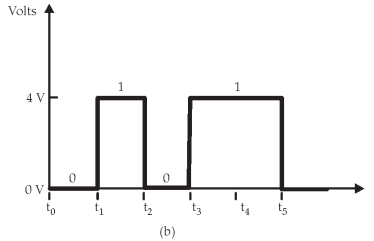
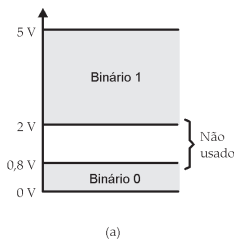


Figura 4 - (a) Valores típicos de tensões em um sistema digital; (b) diagrama de tempo de um sinal digital típico. - (Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer)

5. Circuitos Digitais/Circuitos Lógicos

Os circuitos digitais são também conhecidos por circuitos lógicos. Esses circuitos operam com tensões que se encontram em faixas predeterminadas que representam o binário 0 e o binário 1, não importando o valor exato da tensão, figura 5.

A relação entre as entradas e saídas é estabelecida pela lógica.

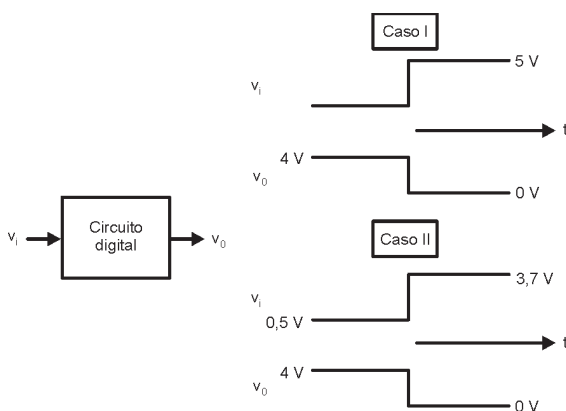


Figura 5 - Um circuito digital responde aos níveis binários das entradas (0 ou 1) e não ao valor exato da tensão. (Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer)

6. Circuitos Integrados Digitais (CIs)

Os CIs tornaram possível a implementação de sistemas digitais complexos menores e mais seguros que os mesmos circuitos implementados com componentes discretos.

As tecnologias de fabricação de CIs usadas para a produção de CIs digitais mais comuns são: **TTL** (lógica transistor-transistor - TBJ), **CMOS** (complementary metal-óxido-semicondutor - MOSFET), **NMOS** e **ECL**.

7. Circuitos Integrados Digitais (CIs)

As duas formas básicas de transferência de informação digital são: paralela (todos os bits são transferidos simultaneamente) e serial (um bit transferido de cada vez), figura 6.

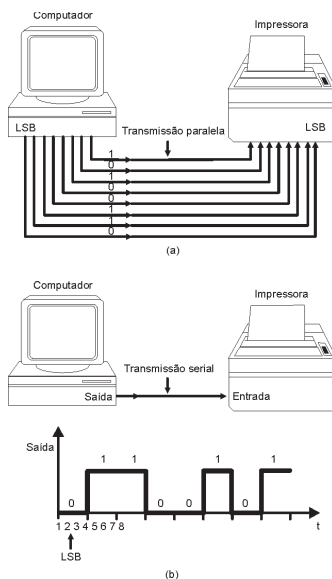
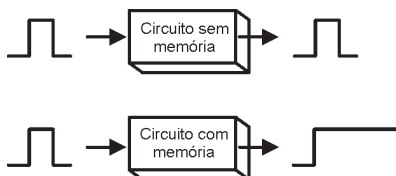


Figura 6 - (a) a transmissão paralela usa uma linha de conexão por bit, e todos os bits são transmitidos simultaneamente; (b) a transmissão serial usa apenas uma linha de sinal, na qual os bits são transmitidos serialmente (um de cada vez). (Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer)

8. Circuitos Integrados Digitais (CIs)

Em circuitos digitais, certos tipos de dispositivos e cir-

cuitos possuem memória. Quando uma entrada é aplicada em um circuito desse tipo, a saída muda de estado e se mantém nesse estado mesmo após a retirada do sinal de entrada, o que não ocorre com um circuito que não possui propriedade de memória, figura 7.



*Figura 7 - Comparação entre as operações com e sem memória.
(Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer)*

9. Computadores Digitais

Um computador é um sistema de hardware que realiza operações aritméticas, manipula dados e toma decisões, mediante um conjunto de instruções denominado **programa**.

As principais partes de um computador são as unidades de entrada, de controle, memória, lógica/aritmética e de saída, figura 8.

A combinação das unidades lógica/aritmética e de controle constitui a CPU (unidade central de processamento) denominada de **microprocessador**.

Existe um tipo de microcomputador mais especializado denominado **microcontrolador**, especialmente projetado para aplicações de controle dedicado.

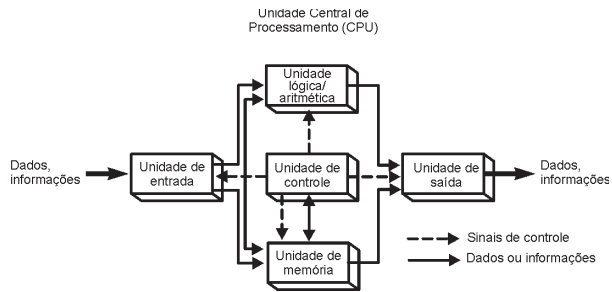


Figura 8 - Diagrama funcional de um computador digital. (Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer)

EXERCÍCIOS PROPOSTOS:

Questões para Revisão (**Livro Texto**):

Seção 1- 1 (questões pág. 3)
Seção 1- 2 (questões pág. 5)
Seção 1- 3 (questões pág. 9)
Seção 1- 5 (questões pág. 12)
Seção 1- 6 (questões pág. 12)
Seção 1- 8 (questões pág. 13)
Problemas (**Livro Texto**):pág. 17

CONSULTAS RECOMENDADAS:

Livro Texto: Sistemas Digitais: Princípios e Aplicações, 8ª
Edição - Ronald J. Tocci e Neal S. Widmer – São Paulo:
Prentice Hall, 2003 - Capítulo 1.

MÓDULO 2

Sistemas de Numeração e Códigos

RESUMO

Há muitos sistemas de numeração em uso na tecnologia digital. Os mais comuns são os sistemas decimal, binário, octal e hexadecimal.

O sistema decimal é o mais importante, porém não é conveniente para ser implementado em sistemas digitais. Os sistemas de numeração octal e hexadecimal são usados em sistemas digitais e computadores como uma alternativa eficiente de representação de quantidades binárias.

OBJETIVO

Em sistemas digitais, diferentes sistemas de numeração podem ser usados ao mesmo tempo, de modo que o entendimento das operações requer a habilidade de converter um número de um sistema para outro. O objetivo é mostrar como realizar essas conversões, apresentando também alguns dos códigos binários que serão usados para representar vários tipos de informações.

CONTEÚDO

2.1 Introdução

O decimal é o mais importante dos sistemas numéricos. Ele está fundamentado em certas regras que são a base de formação para qualquer outro sistema.

Além do sistema **decimal**, que apresenta 10 algarismos distintos de 0 a 9, existe o **binário**, o **octal** e o **hexadecimal**. O sistema binário e o hexadecimal são muito importantes nas áreas de técnicas digitais e informática.

O sistema binário, por sua vez, apresenta somente 2 algarismos (0 e 1), com os quais é possível representar qualquer quantidade, até mesmo números fracionários. No sistema octal existem 8 algarismos que vão de 0 a 7. Para representar o sistema hexadecimal são utilizados 10 algarismos e as 6 primeiras letras do alfabeto e, desta forma, tem-se: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Observando a formação dos infinitos números do sistema decimal é possível aprender as regras de formação dos demais sistemas numéricos. Para conceber a formação do sistema decimal basta observar o hodômetro (marcador de quilômetro) de um automóvel. Quando a “rodinha” das unidades comuta de 9 para 0, um pino nessa rodinha força a rodinha das dezenas a avançar de 1. Assim ocorre sucessivamente formando todos os algarismos.

O mesmo se observa nos demais sistemas. No binário, por exemplo, quando a rodinha da unidade alcança 1 e posteriormente comuta para zero, a rodinha da dezena avança para 1. Pode-se notar que a quantidade de dígitos necessários para representar um número qualquer, no sistema binário, é muito maior quando comparado ao sistema decimal.

A tabela 1 mostra a formação dos algarismos dentro de cada sistema numérico.

Decimal	Binário	Octal	Hexadecimal
000	00000	000	000
001	00001	001	001
002	00010	002	002
003	00011	003	003
004	00100	004	004
005	00101	005	005
006	00110	006	006
007	00111	007	007
008	01000	010	008
009	01001	011	009
010	01010	012	00A
011	01011	013	00B
012	01100	014	00C
013	01101	015	00D
014	01110	016	00E
015	01111	017	00F
016	10000	020	010
017	10001	021	011
018	10010	022	012
019	10011	023	013
020	10100	024	014

Por outro lado, o número decimal 975 pode ser representado da seguinte forma:

$$975 = 900 + 70 + 5 = 9 \times 10^2 + 7 \times 10^1 + 5 \times 10^0$$

Neste exemplo, nota-se que o algarismo menos significativo (5) multiplica a unidade (1 ou 10^0), o segundo algarismo (7) multiplica a dezena (10 ou 10^1) e o mais significativo (9) multiplica a centena (100 ou 10^2). A soma dos resultados irá representar o número.

Pode-se afirmar que, de maneira geral, a regra básica de formação de um número consiste no somatório de cada algarismo correspondente multiplicado pela base (no exemplo o número 10) elevada por um índice conforme o posicionamento do algarismo no número.

Assim, um sistema de numeração genérico pode ser expresso da seguinte forma:

$$N = d_n \times B^n + \dots + d_3 \times B^3 + d_2 \times B^2 + d_1 \times B^1 + d_0 \times B^0$$

Onde:

N é a representação do número na base B ;

d_n é o dígito na posição n ;

B é a base do sistema utilizado e

n é o peso posicional do dígito.

2.2 O Sistema de Numeração Binário

Utiliza somente os algarismos 0 e 1 como códigos, possui base 2. Representa números decimais e letras do alfabeto.

De acordo com a definição de um sistema de numeração genérico, o número binário 1101 pode ser representado da seguinte forma:

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

$$1101_2 = 8 + 4 + 0 + 1 = 13_{10} \text{ (conversão binária } \Rightarrow \text{ decimal)}$$

A vantagem do sistema binário reside no fato de que, possuindo apenas dois dígitos, estes são facilmente representados por uma chave aberta e uma chave fechada ou, um relé ativado e um relé desativado, ou, um transistor saturado e um transistor cortado; o que torna simples a implementação de sistemas digitais mecânicos, eletromecânicos ou eletrônicos.

2.2.1 Conversão de Decimal para Binário

Usa-se o método das divisões sucessivas por 2. Quando o quociente obtido for menor que o divisor deve-se parar a divisão. A resposta é igual ao quociente da última divisão, seguido de todos os restos na ordem inversa. Exemplo: conversão do 47_{10} ® binário

$$\begin{array}{r}
 47 \overline{) 2} \\
 \text{1º resto } \text{---} \text{①} \text{ } 23 \overline{) 2} \\
 \text{2º resto } \text{---} \text{①} \text{ } 11 \overline{) 2} \\
 \text{3º resto } \text{---} \text{①} \text{ } 5 \overline{) 2} \\
 \text{4º resto } \text{---} \text{①} \text{ } 2 \overline{) 2} \\
 \text{5º resto } \text{---} \text{①} \text{ } 1 \text{ --- Último quociente}
 \end{array}$$

$$47_{10} = 101111_2$$

OBS. Usando N bits, podemos representar números decimais na faixa de 0 a $(2^N - 1)$, em um total de 2^N números diferentes.

2.3 O Sistema de Numeração Octal

É um sistema de base 8, ou seja, utiliza 8 algarismos: 0,1,2,3,4,5,6,7. É muitas vezes usado no trabalho com computadores digitais.

Da mesma forma, seguindo a definição de um sistema de numeração genérico, o número octal $24,6_8$ pode ser representado da seguinte forma:

$$24,6_8 = 2 \times (8^1) + 4 \times (8^0) + 6 \times (8^{-1}) = 20,75_{10}$$

(conversão octal => decimal)

2.3.1 Conversão de Decimal para Octal

Utiliza-se, neste caso, o método das divisões sucessivas, lembrando que agora é realizada a divisão por 8, pois 8 é a base do sistema octal. Isso é exemplificado a seguir na conversão do número 92_{10} para octal:

$$\begin{array}{r} 92 \overline{) 8} \\ 1^\circ \text{ resto } \textcircled{4} \quad 11 \overline{) 8} \\ 2^\circ \text{ resto } \textcircled{3} \quad \textcircled{1} \quad \text{Último quociente} \end{array}$$

Assim, seguindo a mesma regra de formação, $92_{10} = 134_8$.

2.3.2 Conversão de Octal para Binário

Como 8 é a terceira potência de 2, pode-se converter octal em binário convertendo-se cada dígito octal em seu equivalente binário de 3 bits.

Considere como exemplo a conversão do número 531_8 para binário:

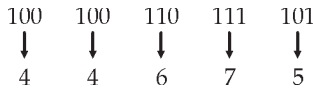
$$\begin{array}{ccc} 5 & 3 & 1 \\ \downarrow & \downarrow & \downarrow \\ 101 & 011 & 001 \end{array}$$

Assim, o número octal 534 é equivalente ao binário 101011001 .

2.3.3 Conversão do Binário para Octal

A conversão de binário para octal é o inverso do procedimento anterior: agrupe os bits de três em três, e converta cada grupo em seu equivalente octal. Se houver necessidade, adicionar zeros a seus extremos.

Para ilustrar, considere a conversão de 100100110111101_2 para octal:



Desta forma, o número $100100110111101_2 = 44675_8$

2.4 O Sistema de Numeração Hexadecimal

O sistema hexadecimal, ou sistema de base 16, é largamente utilizado na área dos microprocessadores e também no mapeamento de memórias em sistemas digitais. Trata-se de um sistema numérico muito importante, aplicado em projetos de software e hardware.

Os algarismos deste sistema são enumerados da seguinte forma: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F**. Nota-se que a letra A representa o algarismo **A**, que por sua vez representa a quantidade dez. O mesmo ocorre para a letra B, que representa o algarismo **B** e a quantidade onze, sucedendo assim até o algarismo **F**, que representa a quantidade quinze.

A conversão do sistema hexadecimal para o sistema decimal pode ser realizada aplicando a definição do sistema de numeração genérico na base 16. Assim, tem-se:

$$N = d_n \times 16^n + \dots + d_2 \times 16^2 + d_1 \times 16^1 + d_0 \times 16^0$$

O processo de conversão é ilustrado nos exemplos abaixo:

$$356_{16} = 3 \times 16^2 + 5 \times 16^1 + 6 \times 16^0 = 768 + 80 + 6 = 854_{10}$$

$$2AF_{16} = 2 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 = 512 + 160 + 15 = 687_{10}$$

2.4.1 Conversão de Decimal para Hexadecimal

Novamente a conversão se faz através de divisões sucessivas pela base do sistema a ser convertido, que no caso é igual a 16. Para exemplificar, o número 1101 na base 10 será convertido para o sistema hexadecimal.

$$\begin{array}{rcl}
 & 1101 & \overline{)16} \\
 1^\circ \text{ resto} & \underline{13} & 68 \overline{)16} \\
 2^\circ \text{ resto} & \underline{4} & \underline{4} \text{ — Último quociente}
 \end{array}$$

Sendo $13_{10} = D_{16}$, tem-se que $1101_{10} = 44D_{16}$

2.4.2 Conversão de Hexadecimal para Binário

Para converter um número hexadecimal em um número binário, converta cada dígito hexadecimal em seu equivalente de 4 bits. Isso está ilustrado a seguir para $C13_{16}$.

C	1	3
↓	↓	↓
1100	0001	0011

Portanto, $C13_{16} = 110000010011_2$

2.4.3 Conversão de Binário para Hexadecimal

Consiste, simplesmente, em fazer o inverso do processo anterior, somente que neste caso são agrupados de 4 em 4 bits da direita para a esquerda. A título de exemplo, será feita a conversão do número binário 100110111110011_2 para hexadecimal.

0100	1101	1111	0011
↓	↓	↓	↓
4	D	F	3

Assim, $100110111110011_2 = 4DF3_{16}$

2.4.4 Conversão de Octal para Hexadecimal e de Hexadecimal para Octal

Converta primeiro para binário, em seguida, converta de binário para o sistema de numeração desejado.

2.5 Números Fracionários

Discutiram-se, até o momento, as diversas formas de conversão de números **inteiros**. Neste tópico, serão mostrados os procedimentos para converter números fracionários.

2.5.1 Conversão de Números Binários Fracionários em Decimais

O método de conversão é obtido observando-se a regra básica de formação de um número fracionário no sistema decimal. Para exemplificar, tem-se o número $10,5_{10}$.

$$10,5_{10} = 1 \times 10^1 + 0 \times 10^0 + 5 \times 10^{-1}$$

Desta forma, para converter o número binário fracionário $101,101_2$ para o sistema decimal, adota-se o mesmo procedimento.

$$101,101_2 = 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 4 + 1 + 0,5 + 0,125$$

$$101,101_2 = 5,625_{10}$$

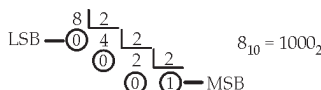
2.5.2 Conversão de Números Decimais Fracionários em Binários

O processo consiste em separar o número decimal na parte inteira e na fracionária. O método das divisões sucessivas é aplicado a parte inteira, conforme estudado anteriormente. Para a parte fracionária aplica-se o método das multiplicações sucessivas até que se atinja zero.

Para exemplificar, será convertido o número decimal $88,375$ em binário.

$$88,375 = 8 + 0,375$$

- Parte inteira:



- Parte Fracionária:



Assim:

$$0,375_{10} = 0,011_2$$

Para completar a conversão basta efetuar a composição da parte inteira com a fracionária:

$$8,375_{10} = 1000,011_2$$

Observação Importante: existem casos em que o método das multiplicações sucessivas encontra novamente os números já multiplicados e o processo entra em um “loop” infinito. Isto equivale a uma dízima periódica. Como exemplo, tem-se:

$$0,810 = (0,1100\ 1100\ 1100\dots)_2$$

2.6 Código BCD

Quando cada dígito de um número decimal for representado pelo seu equivalente binário, o resultado será um código denominado decimal codificado em binário (BCD-binary-coded-decimal). Como um dígito decimal pode ter no máximo o valor 9, são necessários 4 bits para codificar cada dígito (código binário do 9 é 1001).

	8	7	4	decima
	↓	↓	↓	
Ex. 874 D	1000	0111	0100	(BCD)

Obs. O BCD não é outro sistema de numeração, na verdade, é um sistema decimal no qual cada dígito é codificado no seu equivalente binário. Além disso, um número BCD não é o mesmo que um número binário puro. O exemplo a seguir ilustra tal diferença:

$137_{10} = 10001001_2$	(binário)
$137_{10} = 0001\ 0011\ 0111$	(BCD)

2.7 Códigos Alfanuméricos

Um código alfanumérico é um código que usa grupos de bits para representar todos os vários caracteres e funções que fazem parte de um típico teclado de computador. O código ASCII (American Standard Code for Information Interchange) é o mais usado dos códigos alfanuméricos. A tabela 2 mostra uma listagem parcial do código ASCII e as equivalências binária, octal e hexadecimal.

Caractere	ASCII de 7 bits	Octal	Hex	Caractere	ASC de 7 bits	Octal	Hex
A	100 0001	101	41	Y	101 1001	131	59
B	100 0010	102	42	Z	101 1010	132	5A
C	100 0011	103	43	0	011 0000	060	30
D	100 0100	104	44	1	011 0001	061	31
E	100 0101	105	45	2	011 0010	062	32
F	100 0110	106	46	3	011 0011	063	33
G	100 0111	107	47	4	011 0100	064	34
H	100 1000	110	48	5	011 0101	065	35
I	100 1001	111	49	6	011 0110	066	36
J	100 1010	112	4A	7	011 0111	067	37
K	100 1011	113	4B	8	011 1000	070	38
L	100 1100	114	4C	9	011 1001	071	39
M	100 1101	115	4D	blank	010 0000	040	20
N	100 1110	116	4E	-	010 1110	056	2E
O	100 1111	117	4F)	010 1000	050	28
P	101 0000	120	50	+	010 1011	053	2B
Q	101 0001	121	51	\$	010 0100	044	24
R	101 0010	122	52	*	010 1010	052	2A
S	101 0011	123	53)	010 1001	051	29
T	101 0100	124	54	-	010 1101	055	2D
U	101 0101	125	55	/	010 1111	057	2F
V	101 0110	126	56	_	010 1100	054	2C
W	101 0111	127	57	=	011 1101	075	3D
X	101 1000	130	58	<RETURN>	000 1101	015	0D
Y				<LINEFEED>	000 1010	012	0A

Exemplo: A seguinte seqüência de bits, 1000011 1001111 1000011, é uma mensagem codificada em ASCII. Que mensagem é essa?

Solução: .converta cada código de 7 bits no seu equivalente em hexa. O resultado é:

43 4F 43. Depois localize na tabela esses valores em hexa e determine o caractere representado por cada valor. O resultado é: C O C

EXERCÍCIOS PROPOSTOS:

Resolva os problemas 2.1 a 2.23 do **Livro Texto**.

CONSULTAS RECOMENDADAS:

Livro Texto: Sistemas Digitais: Princípios e Aplicações, 8ª
Edição - Ronald J. Tocci e Neal S. Widmer – São Paulo:
Prentice Hall, 2003 - Capítulo 2.

MÓDULO 3

Portas Lógicas e Álgebra Booleana

RESUMO

A álgebra Booleana é uma ferramenta matemática usada na análise e no projeto de circuitos digitais. As operações Booleanas básicas são E, OU e NÃO.

As portas lógicas são circuitos digitais (circuitos elétricos) com uma ou mais tensões de entrada, mas somente uma tensão de saída.

OBJETIVO

Neste módulo serão estudados os circuitos lógicos mais básicos, as portas lógicas, que são os blocos fundamentais a partir dos quais todos os outros circuitos lógicos são construídos. Além disso, será mostrado como a operação de diferentes portas lógicas e circuitos mais complexos, construídos a partir da combinação de portas lógicas, podem ser descritos e analisados usando a álgebra Booleana.

CONTEÚDO

3.1 Introdução

Os sistemas digitais são formados por circuitos lógicos denominados de portas lógicas que, utilizados de forma conveniente, podem implementar todas as expressões geradas pela álgebra de Boole.

Existem três portas básicas (**AND** ("E"), **OR** ("OU") e **NOT** ("NÃO")) que podem ser conectadas de várias maneiras, formando sistemas que vão de simples relógios digitais aos computadores de grande porte.

Pode-se, escrever todas as possíveis combinações para os níveis lógicos presentes nas entradas de um circuito digital na chamada **Tabela da Verdade**, que é definida como um mapa onde se depositam todas as possíveis situações com seus respectivos resultados. O número de combinações

possíveis é igual a 2^N , onde N é o número de variáveis de entrada. A figura 1 mostra exemplos de tabelas-verdade para circuitos lógicos de duas, três e quatro entradas.

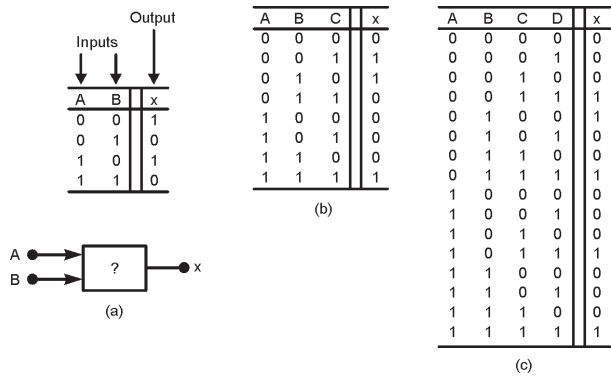


Figura 1 - Exemplos de tabelas-verdade para circuitos de: (a) duas entradas, (b) três entradas e (c) quatro entradas. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

3.2 Operação OR (“OU”) e a Porta OR

A operação OR gera um resultado (saída) 1 sempre que qualquer das entradas for 1. Caso contrário a saída é 0. Sua representação algébrica para duas variáveis de entrada é $X=A+B$, onde se lê: $X=A$ ou B .

Uma porta OU é um circuito lógico que realiza uma operação OU sobre as entradas do circuito. A Fig. 2 ilustra a porta lógica que executa a função **OR** da álgebra de Boole, juntamente com a sua tabela da verdade.

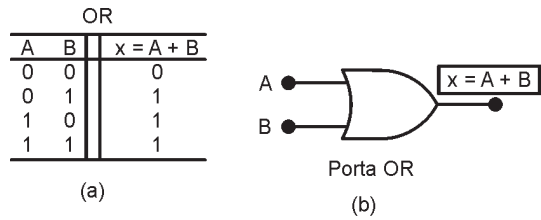


Figura 2 - (a) Tabela-verdade que define a operação OR; (b) símbolo de uma porta OR de duas entradas. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

A fig. 3 mostra o exemplo do uso de uma porta OR em um sistema de alarme.

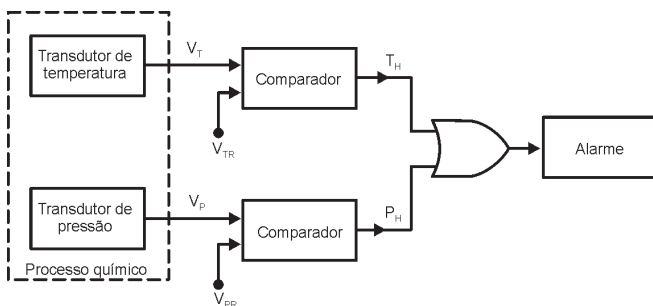


Figura 3 - Exemplo do uso de uma porta OR em um sistema de alarme. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

3.3 Operação AND (“E”) e a Porta AND

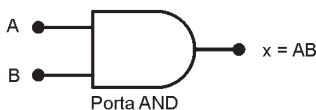
A operação **AND** é realizada da mesma maneira que multiplicação convencional de 1s e 0s, aquela que executa a multiplicação de duas ou mais variáveis booleanas. Sua representação algébrica para duas variáveis é $X=A.B$, onde se lê: $X=A$ e B .

Uma porta AND é um circuito lógico que realiza uma operação AND sobre as entradas do circuito. A Fig. 4 ilustra a porta lógica que executa a função **AND** da álgebra de Boole, juntamente com a sua tabela da verdade.

A saída de uma porta AND será 1 apenas quando todas as entradas forem 1; para todos os outros casos a saída será 0.

AND			
A	B		$x = A \cdot B$
0	0		0
0	1		0
1	0		0
1	1		1

(a)



(b)

Figura 4 - (a) Tabela-verdade que define a operação AND; (b) símbolo de uma porta AND de duas entradas. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

A operação NOT difere das operações OR e AND pelo fato de poder ser realizada sobre uma única variável de entrada. Inverte ou complementa o estado da variável de entrada, ou seja, se a variável estiver em 0, a saída vai para 1, e se estiver em 1 a saída vai para 0. É representada algebricamente da seguinte forma: $X = \bar{A}$, onde se lê: *A barra* ou *NÃO A*.

O inversor é o bloco lógico que executa a função **NOT**. Sua representação simbólica é vista na Fig. 5, juntamente com sua tabela da verdade.

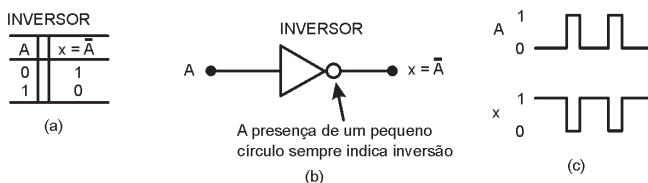


Figura 5 - (a) Tabela-verdade; (b) símbolo para o INVERTOR (circuito NOT); (c) exemplos de formas de ondas. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

Resumo das operações booleanas:

OR	AND	NOT
$0 + 0 = 0$	$0 \cdot 0 = 0$	$\bar{0} = 1$
$0 + 1 = 1$	$0 \cdot 1 = 0$	$\bar{1} = 0$
$1 + 0 = 1$	$1 \cdot 0 = 0$	
$1 + 1 = 1$	$1 \cdot 1 = 1$	

3.5 Função NÃO OU, NOU ou NOR

Esta função é uma composição das funções **OR** e **NOT**, ou seja, é a função **OR** invertida. Sua representação algébrica é $X = \overline{A + B}$, onde o traço indica que ocorrerá uma inversão da soma booleana A+B.

A Fig.6 ilustra o circuito que executa a função NOR da álgebra de Boole, juntamente com sua tabela da verdade.

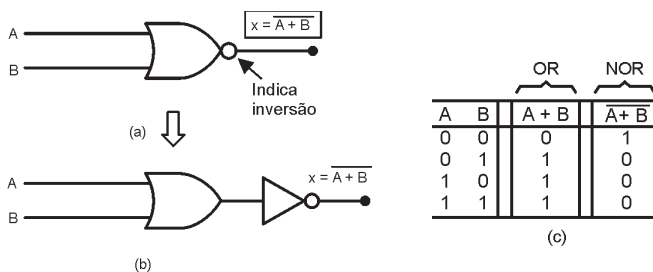


Figura 6 - (a) Símbolo da porta NOR; (b) Circuito equivalente; (c) Tabela-verdade. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

3.6 Função NÃO E, NE ou NAND

Esta função é uma composição das funções **AND** e **NOT**, ou seja, é a função **AND** invertida. Sua representação algébrica é $X = \overline{A \cdot B}$ onde o traço indica que ocorrerá uma inversão do produto booleano $A \cdot B$.

A Fig.7 ilustra o circuito que executa a função NAND da álgebra de Boole, juntamente com sua tabela da verdade.

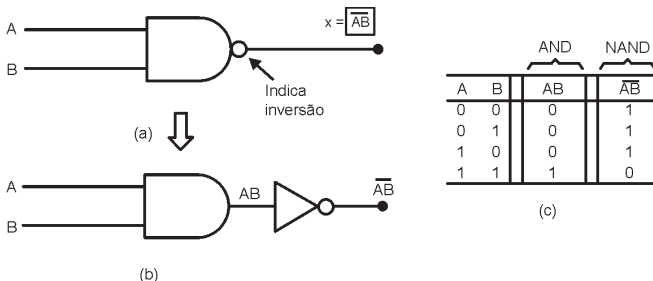


Figura 7 - (a) Símbolo da porta NAND; (b) Circuito equivalente; (c) Tabela-verdade. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

3.7 Função OU EXCLUSIVO "EX-OR"

Esta função, como o próprio nome diz, apresenta saída com valor 1 quando as variáveis de entrada forem diferentes entre si. A notação algébrica que representa a função

OU Exclusivo é $X = A \oplus B$, onde se lê: A **Ou Exclusivo** B.

A Fig. 8 ilustra o circuito que efetivamente realiza a função OU Exclusivo (EX-OR), seu símbolo e sua tabela da verdade. A expressão de saída para esse circuito é:

$$X = \bar{A}.B + A.\bar{B}$$

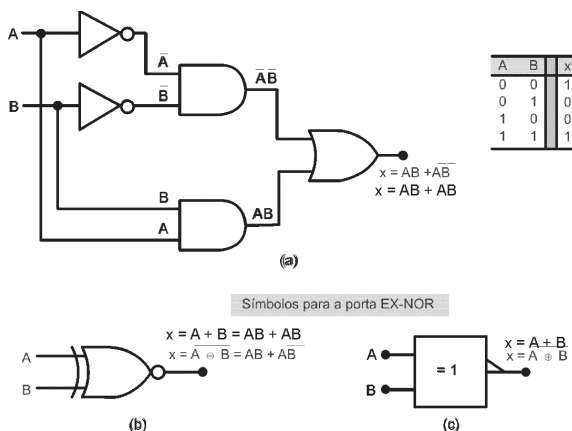


Figura 8 - (a) Circuito Ex-OR; (b) símbolo tradicional para a porta EX-OR; *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

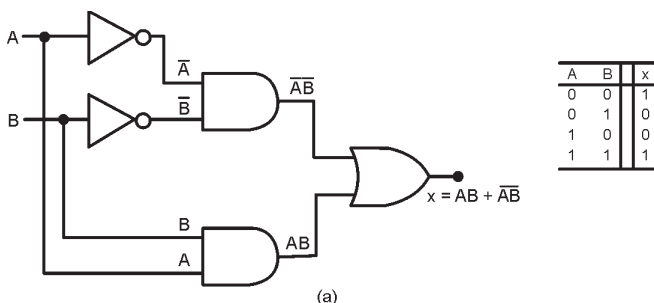
Observação importante: ao contrário dos outros blocos lógicos, cada circuito **EX-OR** admite somente 2 variáveis de entrada.

3.8 Função COINCIDÊNCIA OU NÃO OU EXCLUSIVO "EX-NOR"

Esta função, como seu próprio nome diz, apresenta saída com valor 1 quando houver uma coincidência nos valores das variáveis de entrada, ou seja, é o inverso da operação EX-OR. A notação algébrica que representa a função Coincidência é $X = \overline{A \oplus B}$, onde se lê: A coincidência B.

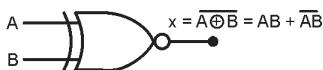
A Fig. 9 ilustra o circuito que efetivamente realiza a função NÃO OU Exclusivo (EX-NOR), seu símbolo e sua tabela da verdade. A expressão de saída para esse circuito é:

$$X = A.B + \bar{A}.\bar{B}$$



(a)

Símbolos para a porta EX-NOR



(b)

Figura 9 - (a) Circuito exclusivo-NOR; (b) símbolo tradicional para a porta EX-NOR; *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

Observação importante: Assim como ocorre com o bloco lógico EX-OR, o circuito COINCIDÊNCIA ou EX-NOR é definido apenas para 2 variáveis de entrada.

3.9 Descrevendo Circuitos Lógicos Algebricamente

Todo o circuito lógico executa uma função booleana e, por mais complexo que seja, é formado pela interligação das portas lógicas básicas. Assim, pode-se obter a expressão booleana que é executada por um circuito lógico qualquer.

Para exemplificar, serão obtidas as expressões que os circuitos da Fig. 10 executam. Observe especialmente o uso de dois conjuntos de parênteses na fig 10(b) indicando qual operação deve ser realizada primeiro.

Quando uma expressão tiver operações AND e OR, a operação AND é realizada primeiro.

Sempre que um INVERSOR estiver presente em um circuito lógico, a expressão para a saída do INVERSOR é igual a expressão de entrada com uma barra sobre ela, conforme pode ser observado na fig.10.

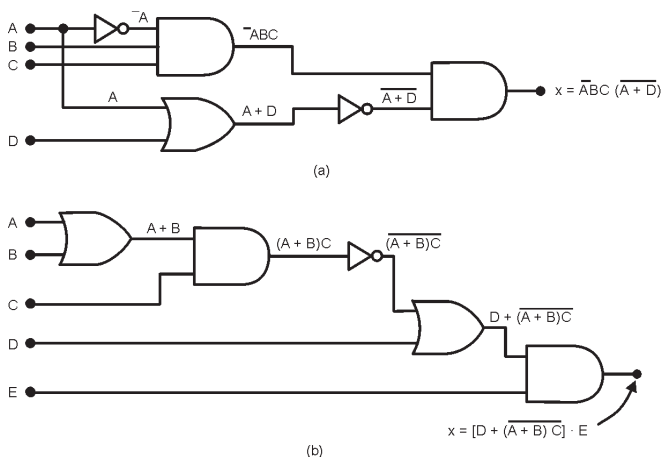


Figura 10 - a) e (b) Circuito lógico e suas expressões Booleanas.
Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer

3.10 Implementando Circuitos A Partir de Expressões Booleanas

É possível desenhar um circuito lógico que executa uma função booleana qualquer, ou seja, pode-se desenhar um circuito a partir de sua expressão característica.

O método para a resolução consiste em se identificar as portas lógicas na expressão e desenhá-las com as respectivas ligações, a partir das variáveis de entrada. Deve-se sempre respeitar a hierarquia das funções da aritmética elementar, ou seja, a solução inicia-se primeiramente pelos parênteses.

Para exemplificar, será obtido o circuito que executa a expressão $x = (A + B)(\bar{B} + C)$:

Essa expressão mostra que os termos $A + B$ e $\bar{B} + C$ são entradas de uma porta AND, e cada um deles é gerado por portas OR independentes. O resultado está demonstrado na fig. 11.

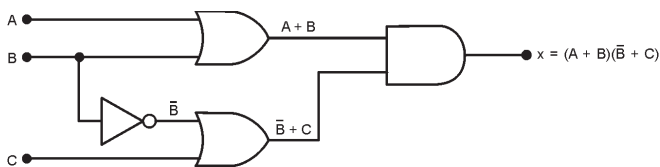


Figura 11 – Construindo um circuito a partir de uma expressão Booleana. Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer

3.11 Tabelas da Verdade Obtidas de Expressões Booleanas

Uma maneira de se fazer o estudo de uma função booleana é a utilização da tabela da verdade. Para extrair a tabela da verdade de uma expressão deve-se seguir alguns procedimentos:

- 1º) Montar o quadro de possibilidades;
- 2º) Montar colunas para os vários membros da equação;
- 3º) Preencher estas colunas com os seus resultados;
- 4º) Montar uma coluna para o resultado final e
- 5º) Preencher esta coluna com os resultados finais.

Para exemplificar este processo, utiliza-se a expressão

$$X = A.\bar{B}.C + A.\bar{D} + \bar{A}.B.D$$

A expressão contém 4 variáveis: A, B, C e D, logo, existem $2^4=16$ possibilidades de combinação de entrada.

Desta forma, monta-se o quadro de possibilidades com 4 variáveis de entrada, três colunas auxiliares, sendo uma para cada membro da expressão, e uma coluna para o resultado final, tabela 1.

Variáveis de entrada				1º membro	2º membro	3º membro	x
A	B	C	D	$A \cdot \bar{B} \cdot C$	$A \cdot \bar{D}$	$\bar{A} \cdot B \cdot D$	
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0	0	1	1
0	1	1	0	0	0	0	0
0	1	1	1	0	0	1	1
1	0	0	0	0	1	0	1
1	0	0	1	0	0	0	0
1	0	1	0	1	1	0	1
1	0	1	1	1	0	0	1
1	1	0	0	0	1	0	1
1	1	0	1	0	0	0	0
1	1	1	0	0	1	0	1
1	1	1	1	0	0	0	0

Tabela 1 – Tabela da Verdade obtida de expressão Booleana:

$$X = A \cdot \bar{B} \cdot C + A \cdot \bar{D} + \bar{A} \cdot B \cdot D$$

3.12 Expressões Booleanas Obtidas de Tabelas da Verdade

Neste item, será estudada a forma de obter expressões e circuitos a partir de tabelas da verdade, sendo este o caso mais comum de projetos práticos, pois, geralmente, necessita-se representar situações através de tabelas da verdade e a partir destas, obter a expressão booleana e conseqüentemente, o circuito lógico.

Para demonstrar este procedimento, será obtida a expressão da seguinte tabela:

A	B	C	S
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(a)

(b)

(c)

(d)

Na tabela, analisa-se onde $X = 1$ e monta-se a expressão adequada.

- Em (a), $X = 1$ se $X = \bar{A} \cdot B \cdot C$

- Em (b), $X = 1$ se $X = A.\bar{B}.C$
- Em (c), $X = 1$ se $X = A.B.\bar{C}$
- Em (d), $X = 1$ se $X = A.B.C$

Para se obter a expressão basta realizar a soma booleana de cada termo acima:

$$X = \bar{A}.B.C + A.\bar{B}.C + A.B.\bar{C} + A.B.C$$

Nota-se que o método permite obter, de qualquer tabela, uma expressão padrão formada sempre pela **soma-de-produtos**. No próximo módulo, será estudado o processo de simplificação de expressões booleanas, possibilitando a obtenção de circuitos reduzidos.

3.13 Teoremas Booleanos

Foi mostrado como a álgebra de Boole pode ser usada na análise de um circuito lógico. Desta forma, deve-se realizar um breve estudo da álgebra de Boole, pois é através de seus postulados, propriedades, teoremas fundamentais e identidades que se efetuam as simplificações. Na álgebra de Boole estão todos os fundamentos da Eletrônica Digital. O primeiro grupo de teoremas é apresentada na fig. 12. Em cada teorema x é uma variável lógica que pode ser 0 ou 1. Para cada teorema é apresentado um circuito lógico que demonstra sua validade.

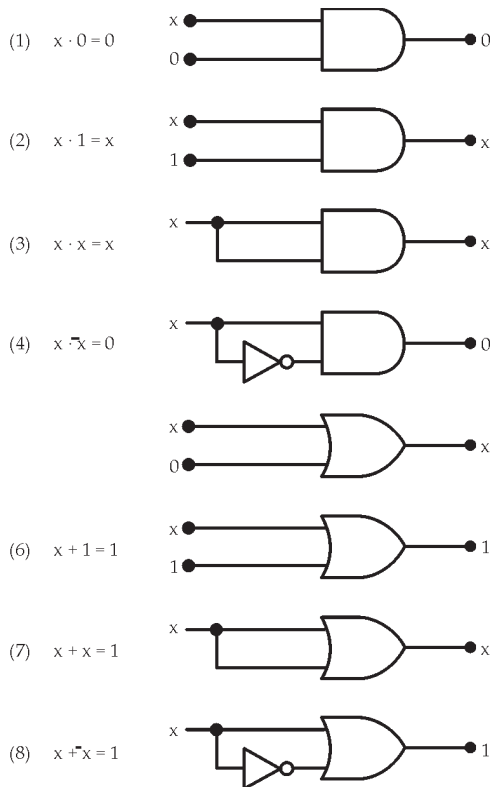


Figura 12 – Teoremas para uma única variável. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

3.13.1 Teoremas com mais de uma variável

- (9) $x + y = y + x$
- (10) $x \cdot y = y \cdot x$
- (11) $x + (y + z) = (x + y) + z$
- (12) $x \cdot (y \cdot z) = (x \cdot y) \cdot z$
- (13a) $x \cdot (y + z) = x \cdot y + x \cdot z$
- (14) $x + x \cdot y = x$
- (15a) $x + \bar{x}y = x + y$
- (15b) $\bar{x} + xy = \bar{x} + y$

OBS. Os teoremas 9 e 10 são chamados leis comutativas;
 Os teoremas 11 e 12 são chamados leis associativas;
 O teorema 13 é a lei distributiva.

3.14 Teoremas de Morgan

São empregados, na prática, para realizar simplificações em expressões booleanas e são utilizados ainda no desenvolvimento de circuitos digitais.

3.14.1 1ª. Teorema de De Morgan

O complemento do produto é igual à soma dos complementos:

$$\overline{(x.y)} = \bar{x} + \bar{y}$$

Pode ainda ser estendido para mais de duas variáveis:

$$\overline{(x.y.w...z)} = \bar{x} + \bar{y} + \bar{w} + \dots + \bar{z}$$

3.14.2 2ª. Teorema de De Morgan

O complemento da soma é igual ao produto dos complementos.

$$\overline{(x + y)} = \bar{x}.\bar{y}$$

Da mesma forma, este teorema pode ser estendido para mais de duas variáveis:

$$\overline{(x + y + w + \dots z)} = \bar{x}.\bar{y}.\bar{w}.....\bar{z}$$

3.15. Universalidade das Portas NAND e NOR

Qualquer expressão pode ser implementada usando combinações de portas OR, AND e INVERSORES. Entretanto, as portas NAND E NOR podem ser usadas para implementar qualquer expressão Booleana básica, figura 15 e figura 16, respectivamente. Em ambas as figuras os pequenos círculos representam a operação de inversão.

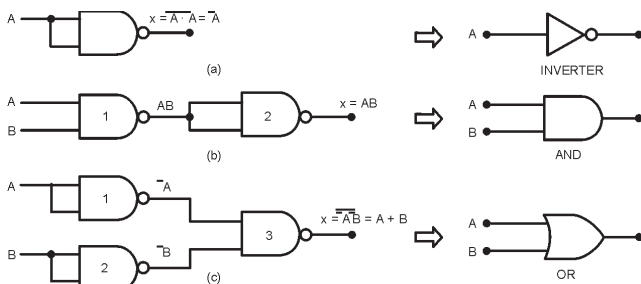


Figura 15 – As portas NAND podem ser usadas para implementar qualquer operação booleana. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

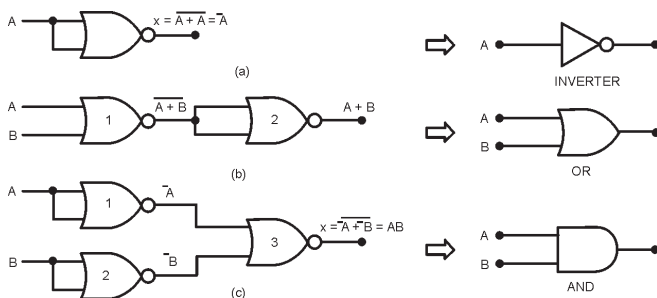


Figura 16 – As portas NOR podem ser usadas para implementar qualquer operação booleana. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

3.16. Simbologia Alternativa para Portas Lógicas (Dualidade)

O símbolo alternativo de cada porta é obtido a partir do símbolo padrão, aplicando o conhecido Teorema da Dualidade:

- Inverta cada entrada e cada saída do símbolo padrão.
- Troque todas as portas AND por Or e todas as portas OR por AND.

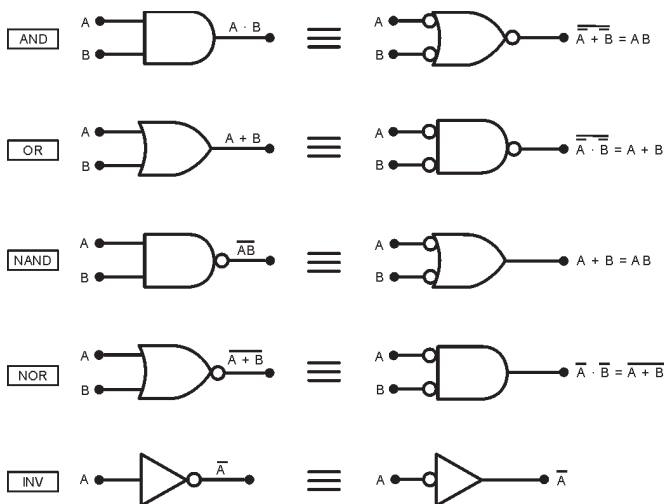


Figura 17 – Símbolo-padrão e alternativos para varias portas lógicas. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

OBS. Quando uma linha de entrada ou saída em um símbolo lógico não tem um pequeno círculo, diz-se que ela é ativa-ALTO. Quando uma linha de entrada ou saída tem um pequeno círculo, diz-se que ela é ativa-BAIXO. A figura 18 mostra o símbolo-padrão para uma porta NAND que tem um pequeno círculo na saída e nenhum círculo nas entradas.

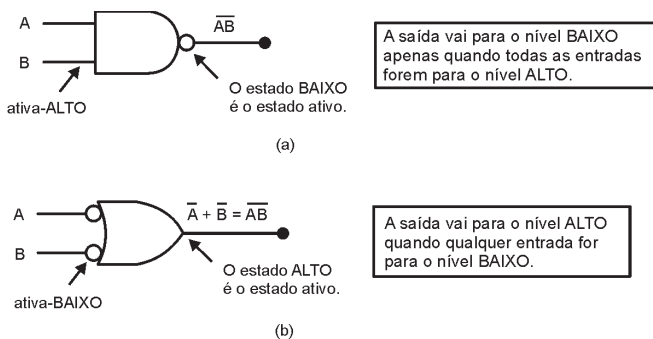


Figura 18 – Porta NAND. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

3.17. Simbologia do Padrão IEEE/ANSI

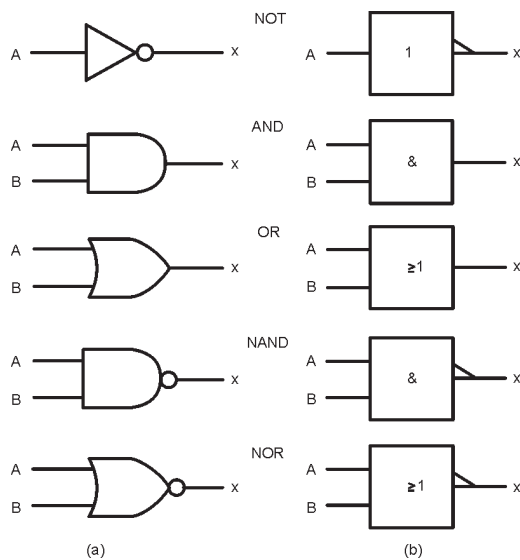


Figura 19 – Símbolos lógicos-padrão: (a) tradicional; (b) IEEE/ANSI
 Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer

EXERCÍCIOS PROPOSTOS:

Resolva os problemas 3.1 a 3.21 e 3.43 a 3.45 do capítulo 3 do **Livro Texto**.

CONSULTAS RECOMENDADAS:

Livro Texto: Sistemas Digitais: Princípios e Aplicações, 8ª
 Edição - Ronald J. Tocci e Neal S. Widmer – São Paulo:
 Prentice Hall, 2003 - Capítulo 3.

MÓDULO 4

Circuitos Lógicos Combinacionais - Parte1

RESUMO

Circuitos combinacionais são aqueles em que a saída depende única e exclusivamente das combinações entre as variáveis de entrada, portanto não possuem a característica de memória. Esses circuitos podem ser projetados através de técnicas simples. No projeto de um circuito lógico uma das preocupações é a simplificação do circuito, visando minimizar a quantidade de portas lógicas do circuito.

OBJETIVO

Neste módulo estudaremos a simplificação de circuitos lógicos. Dois métodos serão usados: a álgebra de Boole e a técnica de mapeamento (Mapa de Karnaugh). Além disso, será apresentado um procedimento completo de projeto, o qual, a partir de um determinado conjunto de requisitos permite a obtenção de um circuito que atenda eficientemente tais requisitos.

CONTEÚDO

4.1 Introdução

No módulo anterior, os circuitos lógicos foram tratados sem a preocupação da simplificação, o que na prática deve ser realizado visando minimizar a quantidade de portas lógicas do circuito. A figura 1(a) mostra que um circuito pode ser simplificado, resultando em um circuito mais simples, figura 1(b), capaz de realizar a mesma lógica.

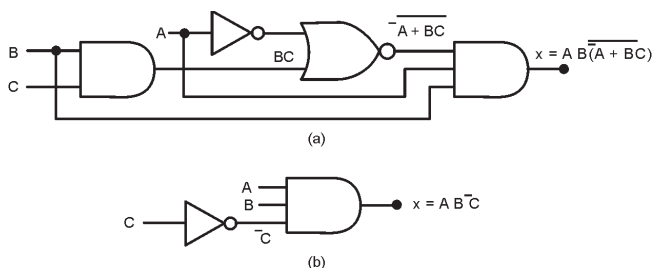


Figura 1 – (a) Circuito lógico sem simplificação - (b) circuito simplificado. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

Na sequência serão apresentados dois métodos de simplificação de circuitos lógicos: simplificação Algébrica e simplificação por Mapas de Karnaugh. Esses métodos requerem que a expressão esteja na forma de **soma-de-produtos**, já abordada no módulo anterior.

4.1 Simplificação Algébrica de Expressões Booleanas

Utilizando os conceitos da álgebra de boole estudados no módulo anterior é possível simplificar expressões e conseqüentemente circuitos.

Exemplo1: Simplifique as expressões abaixo usando a álgebra booleana

OBS. Para facilitar, primeiramente a expressão de ser colocada na forma de soma-de-produtos. Em seguida, observa-se se termos produto têm fatores comuns e realiza-se a fatoração desses termos.

$$a) S = ABC + \overline{A}\overline{B}(\overline{\overline{A}\overline{C}})$$

$$b) S = \overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}C + ABC$$

$$c) S = \bar{A}C(\overline{\bar{A}BD}) + \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C$$

$$d) S = \bar{A}\bar{B}C + \bar{A}BD + \bar{C}\bar{D}$$

Exemplo2: Simplifique o circuito mostrado na figura 1(a) de forma a obter o circuito da figura 1(b).

Exemplo 3: Desenhe o circuito lógico da expressão booleana dada abaixo usando apenas portas NAND. Prove que os dois circuitos são equivalentes. Em seguida simplifique a expressão usando a álgebra booleana e desenhe o circuito resultante.

$$X = \bar{A}.B.C + A.\bar{B}.C + A.B.\bar{C} + A.B.C$$

4.2 Simplificação de Expressões Booleanas Através dos Mapas de Veitch-Karnaugh

Quando são utilizados os teoremas e postulados Booleanos para simplificação de expressões lógicas não se pode afirmar, em vários casos, que a equação resultante está na sua forma minimizada.

Existem métodos de mapeamento das expressões lógicas que possibilitam a simplificação de expressões de N variáveis. O diagrama ou mapa de Karnaugh é um destes métodos e permite a simplificação mais rápida dos casos extraídos diretamente de tabelas da verdade, obtidas de situações quaisquer. Serão estudados os diagramas para 2, 3 e 4 variáveis.

A figura 2 mostra três exemplos de mapas de Karnaugh, para duas, três e quatro variáveis. Cada linha da tabela da verdade possui uma região definida no diagrama de Veitch-Karnaugh. Essas regiões são os locais onde devem ser colocados os valores que a expressão assume (0 ou 1) nas diferentes possibilidades.

Observe que de uma posição para outra no mapa, apenas uma variável muda de estado.

A	B	X
0	0	1 → $\bar{A}\bar{B}$
0	1	0
1	0	0
1	1	1 → AB

$$\left\{ x = \bar{A}\bar{B} + AB \right\}$$

\bar{B}	B
\bar{A}	1
A	0
	1

(a)

A	B	C	X
0	0	0	1 → $\bar{A}\bar{B}\bar{C}$
0	0	1	1 → $\bar{A}\bar{B}C$
0	1	0	1 → $\bar{A}B\bar{C}$
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1 → $AB\bar{C}$
1	1	1	0

$$\left\{ x = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + AB\bar{C} \right\}$$

\bar{C}	C
$\bar{A}\bar{B}$	1
$\bar{A}B$	1
AB	1
$\bar{A}\bar{B}$	0

(b)

A	B	C	D	X
0	0	0	0	0
0	0	0	1	1 → $\bar{A}\bar{B}\bar{C}D$
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1 → $\bar{A}\bar{B}C\bar{D}$
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1 → $A\bar{B}\bar{C}D$
1	1	1	0	0
1	1	1	1	1 → $ABCD$

$$\left\{ x = \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + A\bar{B}\bar{C}D + ABCD \right\}$$

$\bar{C}D$	$\bar{C}\bar{D}$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	0
$\bar{A}B$	0	1	0
AB	0	1	1
$\bar{A}\bar{B}$	0	0	0

(c)

Figura 2 - Mapas de Karnaugh e tabelas-verdade para (a) duas, três e (c) quatro variáveis. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

A expressão simplificada é obtida do diagrama, cujo método consiste em agrupar as regiões onde $X=1$ no menor número possível de agrupamentos. Os termos que não puderem ser agrupados devem ser considerados isoladamente.

Os agrupamentos devem ser realizados considerando os grupos de 1s adjacentes, que podem resultar em pares (dois 1s adjacentes), quadras (quatro 1s adjacentes) ou octetos (oito 1s adjacentes). As simplificações resultantes são:

- Um termo isolado não admite simplificação;
- Um par elimina uma variável e seu complemento;
- Uma quadra elimina duas variáveis e seus complementos;
- ~ Um Octeto, elimina três variáveis e seus complementos.

As figuras 3, 4 e 5 mostram mapas de Karnaugh que têm agrupamentos de pares, quadras e octetos, respectivamente, bem como as variáveis eliminadas em cada situação.

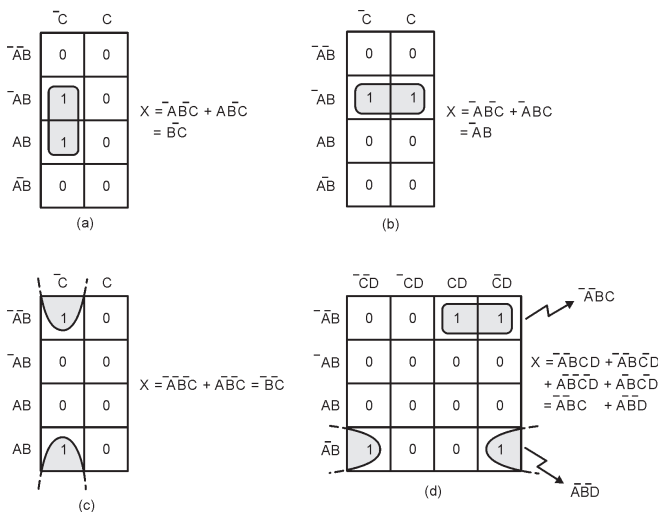


Figura 3 - Exemplo de agrupamentos de pares de 1s adjacentes e as simplificações resultantes. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

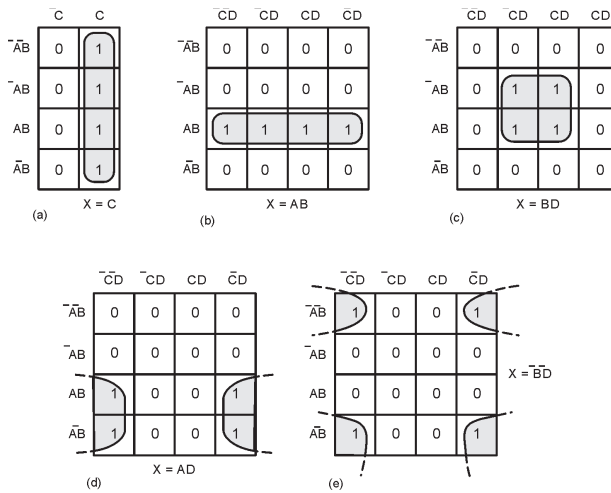


Figura 4 - Exemplo de agrupamentos de quatro 1s adjacentes e as simplificações resultantes. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

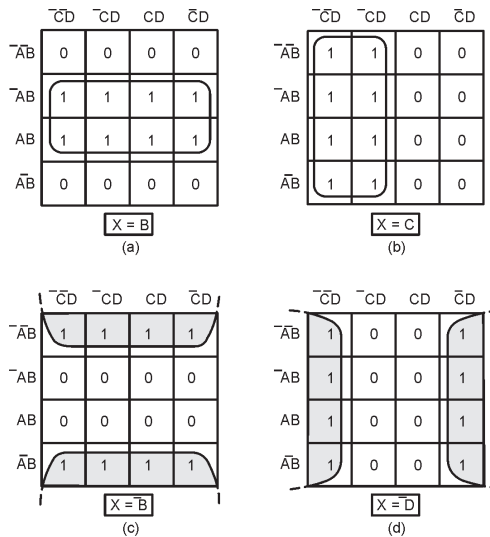


Figura 5 - Exemplo de agrupamentos de oito 1s adjacentes e as simplificações resultantes. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

Procedimento:

- Construa o Mapa de Karnaugh e coloque os 1s nos quadros que correspondem aos 1s na tabela-verdade. Coloque 0s nos outros quadros;
- Primeiro circunde os octetos, depois as quadras e por ultimo os pares;
- Se houver 1s isolados marque-os;
- Para finalizar, somam-se as expressões referentes aos agrupamentos.

A figura 6 mostra as simplificações para três mapas de Karnaugh diferentes.

Obs 1. Pode ocorrer a sobreposição de grupos: pode-se usar o mesmo 1 mais de uma vez. O um que faz parte de um octeto pode também fazer parte de uma quadra ou par. Isto ajuda a simplificar mais a equação, conforme mostra a figura 6(a) e 6(b).

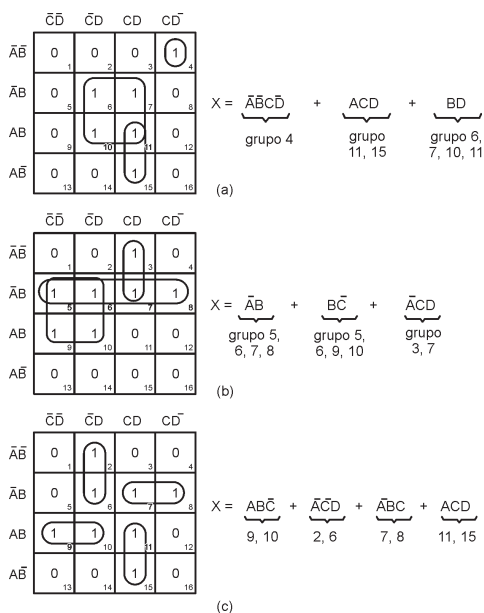


Figura 5 - Exemplos de simplificações para três mapas de Karnaugh diferentes. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

Obs 2. Enrolando o Mapa: o exemplo a seguir mostra como o mapa poder ser enrolado de forma que o lado esquerdo encoste no lado direito, os dois pares agora formam uma quadra, simplificando ainda mais a expressão. Nesse caso resulta: $X = \bar{A}B + \bar{C}$.

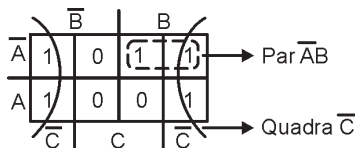


Figura 6 – Enrolando o mapa de Karnaugh

4.2.1 Diagramas com Condições Que não Importam:

Condição irrelevante ou que não importa (x) ocorre quando a saída pode assumir 0 ou 1 indiferentemente, para uma dada situação de entrada. Na prática, esta condição ocorre principalmente pela impossibilidade da situação de entrada acontecer.

Desta forma, os valores irrelevantes da tabela da verdade devem ser transportados para o diagrama de Karnaugh. Assim, para efetuar as simplificações, a condição irrelevante x pode ser utilizada para completar um agrupamento, minimizando a expressão característica e conseqüentemente o circuito lógico.

Por outro lado, se a condição irrelevante x representar um termo isolado, deverá ser descartada.

A figura 7 mostra uma tabela-verdade com duas condições irrelevantes e o correspondente mapa de Karnaugh com as simplificações resultantes. Observe que para a condição que não importa a saída pode ser considerada 0 ou 1 de forma que se obtenha o melhor agrupamento possível e a máxima simplificação.

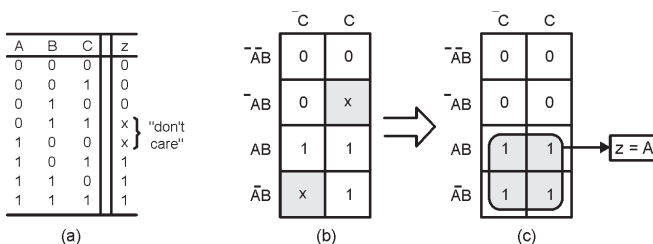


Figura 7 - Condições que não importam devem ser alteradas para 0 ou para 1 de forma a gerar agrupamentos no mapa k que produzam a expressão mais simples. Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer

4.2.2 Casos Que Não Admitem Simplificações

As funções **OU EXCLUSIVO (EX-OR)** e **COINCIDÊNCIA (EX-NOR)** são exemplos de casos que não admitem simplificações, pois suas equações características estão minimizadas, como ilustra a figura abaixo.

$$X = A \oplus B = \bar{A}B + A\bar{B} \quad X = \overline{A \oplus B} = \bar{A}\bar{B} + AB$$

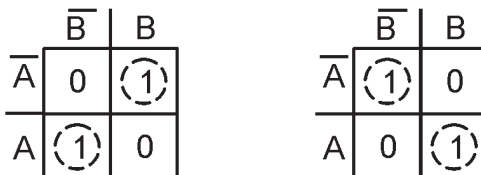


Figura 8 – Expressões (a) EX-OR e (b) EX-NOR não admitem simplificações

Como pode ser observado, em cada diagrama existem dois termos isolados que são, portanto, as próprias expressões de entrada.

No caso de 3 variáveis, as expressões são:

$$X = \underline{A \oplus B \oplus C}$$

$$X = \underline{A \oplus B \oplus C}$$

Para montar a tabela da verdade deve-se primeiramente efetuar as operações entre 2 das variáveis e, com o resultado obtido, efetuar a operação com a terceira variável. Este

processo se deve ao fato de as funções **OU EXCLUSIVO** e **COINCIDÊNCIA** não serem válidas para mais de 2 variáveis de entrada.

4.3 Projeto de Circuitos Lógicos Combinacionais

Quando o nível de saída desejado de um circuito lógico é dado para todas as condições de entrada possíveis, pode-se apresentar todos os resultados em uma tabela-verdade. Como já abordado no módulo 3, a partir da tabela-verdade é possível determinar a expressão de saída do circuito na forma de soma-de-produtos. O circuito resultante pode ser facilmente implementado usando portas AND, OR e INVERSOES. Entretanto, a expressão pode ser simplificada a fim de obter um circuito mais simples. Esses são os procedimentos de projeto para a construção de circuitos lógicos e estão resumidos a seguir:

- 1º. Passo:** Analisar o problema;
- 2º. Passo:** Estabelecer convenções;
- 3º. Passo:** Montar a tabela-verdade;
- 4º. Passo:** Obter a expressão na forma-de-soma produtos;
- 5º. Passo:** Simplificar a expressão de saída;
- 6º. Passo:** Implementar o circuito para a expressão final.

O circuito lógico, obtido seguindo os procedimentos acima, pode apresentar diversas variáveis de entrada e possuir diversas saídas (fig. 9), conforme o projeto.

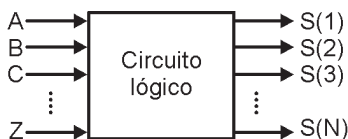
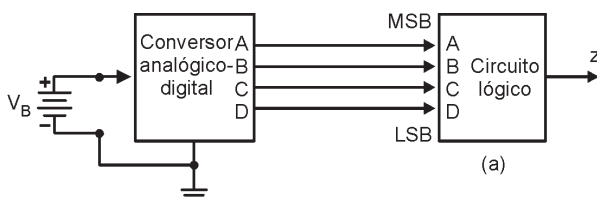


Figura 9 – Circuito pode apresentar várias variáveis de entrada e várias saídas.

Exemplo 1

- Análise do problema:**

A figura abaixo mostra um conversor analógico-digital sendo usado para monitorar uma tensão dc de uma bateria de 12 V de uma espaçonave em órbita. A saída do conversor é um número binário de quatro bits, ABCD, que corresponde à tensão da bateria em degraus de 1V, sendo a variável A o MSB (bit mais significativo). As saídas binárias do conversor são as entradas de um circuito que gera uma saída em nível ALTO que a tensão da bateria for maior que 6V. Projete esse circuito lógico.



- Estabelecer Convenções:**

a) $Z = 1$ sempre que o valor binário for maior que $0110_2 = 6_{10}$, ou seja, quando a tensão da bateria for maior que 6V

- Montar a Tabela da Verdade:**

	A	B	C	D	z
(0)	0	0	0	0	0
(1)	0	0	0	1	0
(2)	0	0	1	0	0
(3)	0	0	1	1	0
(4)	0	1	0	0	0
(5)	0	1	0	1	0
(6)	0	1	1	0	0
(7)	0	1	1	1	1ⓐ $\bar{A}BCD$
(8)	1	0	0	0	1ⓑ $AB\bar{C}\bar{D}$
(9)	1	0	0	1	1ⓒ $AB\bar{C}D$
(10)	1	0	1	0	1ⓓ $AB\bar{C}\bar{D}$
(11)	1	0	1	1	1ⓔ $ABCD$
(12)	1	1	0	0	1ⓖ $AB\bar{C}\bar{D}$
(13)	1	1	0	1	1ⓗ $AB\bar{C}D$
(14)	1	1	1	0	1ⓓ $ABCD$
(15)	1	1	1	1	1ⓔ $ABCD$

- Escrever a expressão de saída na forma de soma-de-produtos:

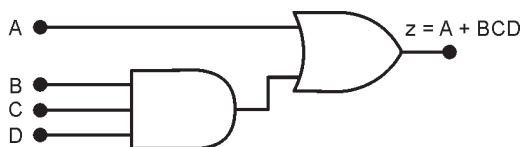
$$Z = \bar{A}BCD + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D} + A\bar{B}C\bar{D} + A\bar{B}CD + AB\bar{C}\bar{D} + AB\bar{C}\bar{D} + ABC\bar{D} + ABCD$$

- Simplificar a expressão de saída:

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	0	0	0
$\bar{A}B$	0	0	1	0
AB	1	1	1	1
$A\bar{B}$	1	1	1	1

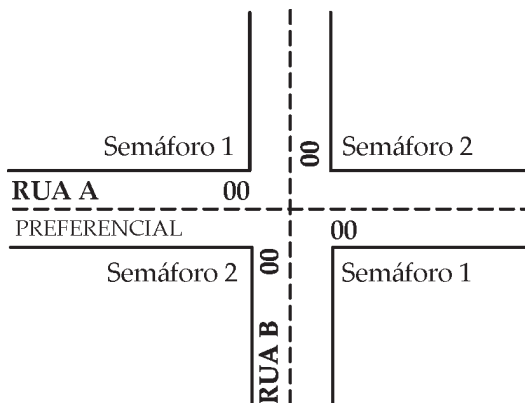
$Z = A + BCD$

- Implementar o circuito para a expressão final



Exemplo 2:

Instalação de um sistema automático de semáforo no cruzamento das ruas A (preferencial) e B.



- 1) Quando houver carros transitando somente na Rua B, o semáforo 2 deverá permanecer verde.
- 2) Quando houver carros transitando somente na Rua A, o semáforo 1 deverá permanecer verde.

- 3) Quando houver carros transitando nas Ruas A e B, o semáforo da Rua A deverá estar verde, pois é preferencial.

• **Estabelecer Convenções:**

- a) Existência de carro na Rua A: $A = 1$
- b) Não existência de carro na Rua A: $A = 0$
- c) Existência de carro na Rua B : $B = 1$
- d) Não existência de carro na Rua B: $B = 0$
- e) Verde do sinal 1 aceso: $V_1 = 1$
- f) Verde do sinal 2 aceso: $V_2 = 1$
- g) Quando $V_1 = 1$
 - Vermelho do semáforo 1 apagado: $V_{m1} = 0$
 - Verde do semáforo 2 apagado: $V_2 = 0$
 - Vermelho do semáforo 2 aceso: $V_{m2} = 1$
- h) Quando $V_2 = 1$? $V_1 = 0$, $V_{m2} = 0$, $V_{m1} = 1$

• **Montar a Tabela da Verdade:**

Entrada		Saídas			
A	B	V_1	V_{m1}	V_2	V_{m2}
0	0	X	X	X	X
0	1	0	1	1	0
1	0	1	0	0	1
1	1	1	0	0	1

- **Escrever a expressão das saídas na forma de soma-de-produtos:**

$$V_1 = V_{m2} = A\bar{B} + AB$$

$$V_{m1} = V_2 = \bar{A}B$$

- **Obter a Expressão Simplificada:**

	\bar{B}	B
\bar{A}	X	0
A	1	1

Mapa para V_1

	\bar{B}	B
\bar{A}	X	1
A	0	0

Mapa para V_{m1}

	\bar{B}	B
\bar{A}	X	1
A	0	0

Mapa para V_2

	\bar{B}	B
\bar{A}	X	0
A	1	1

Mapa para V_{m2}

Pela Tabela da Verdade ou pelo Mapa de Karnaugh pode-se observar que as expressões de V_1 e V_{m2} são idênticas, o mesmo ocorrendo com V_2 e V_{m1} . Assim, as expressões simplificadas são:

$$V_1 = V_{m2} = A \quad \text{e} \quad V_2 = V_{m1} = \bar{A}$$

- **Implementar o circuito para a expressão final**

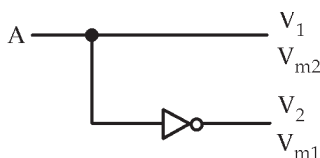


Figura 13 – Circuito resultante do Exemplo 2.

Conclui-se, observando o circuito lógico, que a presença de carro na rua preferencial ($A=1$) acarreta o acendimento do verde do semáforo 1 e o vermelho do semáforo 2 e, devido à ação do inversor, o apagamento do verde do semáforo 2 e vermelho do sinal 1. A ausência de carros nesta via ($A=0$), causa a condição contrária, o que possibilita a abertura da via secundária. Observa-se, ainda, que a variável B foi eliminada das expressões no processo de simplificação, devi-

do às situações consideradas no projeto. Assim, para a realização deste circuito, basta simplesmente colocar um sensor de presença de veículos na Rua A e utilizar uma porta inversora.

Exercício 1:

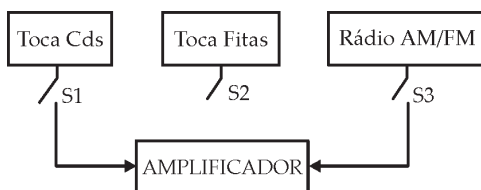
Deseja-se utilizar um único amplificador para ligar três aparelhos: um toca CDs, um toca fitas e um rádio AM/FM. O circuito lógico deverá ligar os aparelhos obedecendo as seguintes prioridades:

1a prioridade: Toca CDs

2a prioridade: Toca fitas

3a prioridade: Rádio AM/FM

Projeto o circuito lógico.



Exercício 2:

Uma empresa deseja adotar um sistema de prioridade nos seus intercomunicadores. As prioridades são:

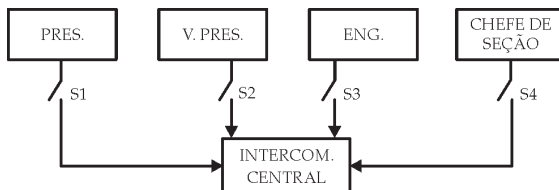
1a prioridade: Presidente

2a prioridade: Vice-presidente

3a prioridade: Engenharia

4a prioridade: Chefe de seção

Projete o circuito lógico.



4.3.1 Circuitos Gerador e Verificador de Paridade

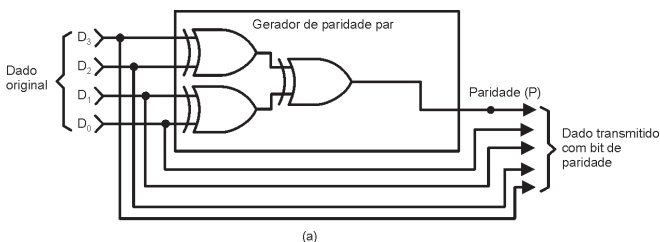
Quando uma informação é transmitida de um dispositivo para outro, há a possibilidade de ocorrência de erros quando o receptor não recebe uma informação idêntica àquela que foi enviada pelo transmissor. A principal causa de um erro de transmissão é o ruído elétrico. Por isso, muitos sistemas digitais utilizam algum método de detecção de erros. Uma das técnicas mais simples e mais usadas para a detecção de erros é o método de paridade.

As portas EX-OR “OU EXCLUSIVO” são ideais para testar a paridade de palavras digitais, já que sua saída será ALTA só com número ÍMPAR de entradas em nível ALTO, e BAIXA se o número de 1s nas entradas for PAR.

- **Paridade Par:** significa que uma determinada palavra digital tem um número par de bits “1”. Ex: 11110011 \Rightarrow SAÍDA DA PORTA OU EXCLUSIVO = 0
- **Paridade Ímpar:** significa que uma determinada palavra digital tem um número ímpar de bits “1”. Ex: 11110001 \Rightarrow SAÍDA DA PORTA OU EXCLUSIVO = 1

Assim as portas EX-OR podem ser utilizadas para implementar geradores e verificadores de paridade.

Na figura 16 (a), o conjunto de dados a serem transmitidos são aplicados ao circuito gerador de paridade, que produz um bit de paridade par, P, em sua saída. Esse bit de Paridade é transmitido para o receptor juntamente com os bits do dado original, totalizando cinco bits. Na figura 16 (b), esses cinco bits entram no circuito verificador de paridade do receptor, o qual gera uma saída de erro, E, que indica se ocorreu, ou não, um erro em um único bit.



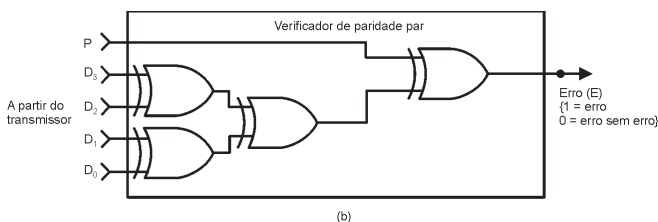


Figura 16 -. Portas EX-OR utilizadas para implementar um gerador de paridade (a) e um verificador de paridade (b) para um sistema que usa paridade par. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

EXERCÍCIOS PROPOSTOS:

Resolva os problemas 4.1 a 4.9 e 4.16 a 4.19, 4.25 e 4.26, 4.28 e 4.29 do capítulo 4 do **Livro Texto**.

CONSULTAS RECOMENDADAS:

Livro Texto: Sistemas Digitais: Princípios e Aplicações, 8ª
Edição - Ronald J. Tocci e Neal S. Widmer – São Paulo:
Prentice Hall, 2003 - Capítulo 4.

MÓDULO 5

Circuitos Lógicos Combinacionais – Parte II

RESUMO

Circuitos combinacionais são aqueles em que a saída depende única e exclusivamente das combinações entre as variáveis de entrada, portanto não possuem a característica de memória. Esses circuitos podem ser projetados através de técnicas simples. No projeto de um circuito lógico uma das preocupações é a simplificação do circuito, visando minimizar a quantidade de portas lógicas do circuito.

OBJETIVO

Dando continuidade aos circuitos combinacionais, esse módulo tratará do projeto de circuitos codificadores e decodificadores.

CONTEÚDO

Codificadores, decodificadores e os circuitos aritméticos são circuitos combinacionais empregados principalmente na arquitetura interna de circuitos integrados e, ainda, em sistemas digitais.

Para a construção dos codificadores e decodificadores serão apresentados os códigos digitais mais conhecidos e de maior utilidade.

5.1 Códigos

São vários os códigos dentro do campo da eletrônica digital, existindo situações em que a aplicação de um é mais vantajoso em relação a outro.

5.1.1 Código BCD 8421

BCD ou “Binary Coded Decimal” significa uma codificação do sistema binário em decimal. Os termos seguintes 8421 significam os valores dos algarismos num dado número binário e representam respectivamente: 2^3 , 2^2 , 2^1 e 2^0 .

Decimal	BCD 8421			
	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

O número de bits de um código é o número de dígitos binários que este possui. Desta forma, o código BCD 8421 é um código de 4 bits.

5.1.2 Código Excesso 3

Este código é composto pela transformação do número decimal em binário, somando-se 3 unidades, ou seja: $010 = 0+3 \text{ unidades} = 0000_{(2)} + 0011_{(2)} = 0011_{(2)}$

Decimal	Excesso 3			
	A	B	C	D
0	0	0	1	1
1	0	1	0	0
2	0	1	0	1
3	0	1	1	0
4	0	1	1	1
5	1	0	0	0
6	1	0	0	1
7	1	0	1	0
8	1	0	1	1
9	1	1	0	0

5.1.3 Código Gray

Sua principal característica é que de um número a outro apenas um bit varia.

Decimal	Gray			
	A	B	C	D
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0

5.1.4 Código de 5 Bits: 2 entre 5

Trata-se de um código que possui sempre dois bits iguais a 1, dentro de 5 bits.

Decimal	2 entre 5				
	A	B	C	D	E
0	0	0	0	1	1
1	0	0	1	0	1
2	0	0	1	1	0
3	0	1	0	0	1
4	0	1	0	1	0
5	0	1	1	0	0
6	1	0	0	0	1
7	1	0	0	1	0
8	1	0	1	0	0
9	1	1	0	0	0

5.1.6 Código de 5 Bits: Johnson

Trata-se de um código que será utilizado na construção do contador Johnson.

Decimal	Johnson				
	A	B	C	D	E
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	1
3	0	0	1	1	1
4	0	1	1	1	1
5	1	1	1	1	1
6	1	1	1	1	0
7	1	1	1	0	0
8	1	1	0	0	0
9	1	0	0	0	0

5.1.7 Código 9876543210

Este código é composto por 10 bits, dentre os quais somente um algarismo vale 1 em cada caso, acendendo assim o algarismo correspondente.

Decimal	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	1	0	0
3	0	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	1	0	0	0	0
5	0	0	0	0	1	0	0	0	0	0
6	0	0	0	1	0	0	0	0	0	0
7	0	0	1	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	0	0	0	0

5.2 Codificadores e Decodificadores

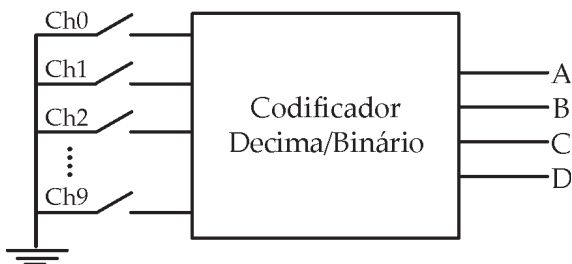
Os codificadores são circuitos combinacionais que possibilitam a passagem de um código conhecido para um desconhecido. Os circuitos decodificadores fazem o inverso, ou seja, passam um código desconhecido para um conhecido.

Equipamentos digitais e alguns sistemas de computação têm seus dados de entrada expressos em decimal, facilitando o trabalho do operador. Entretanto, estes dados são processados internamente em binário e o trabalho de conversão é realizado pelos circuitos codificadores. Os dados já processados são novamente convertidos em decimal, na forma compatível para um mostrador digital apresentar os algarismos. Este trabalho é feito pelos circuitos decodificadores.



5.2.1 Codificador Decimal . Binário

Será desenvolvido o circuito lógico que realiza a codificação de Decimal em Binário (BCD8421). Neste circuito serão utilizadas portas TTL. **Uma das características da família TTL é que os terminais de entrada em vazio (desconectados) são equivalentes a nível lógico 1.**



Chave	A	B	C	D
Ch0	0	0	0	0
Ch1	0	0	0	1
Ch2	0	0	1	0
Ch3	0	0	1	1
Ch4	0	1	0	0
Ch5	0	1	0	1
Ch6	0	1	1	0
Ch7	0	1	1	1
Ch8	1	0	0	0
Ch9	1	0	0	1

Através da tabela conclui-se que:

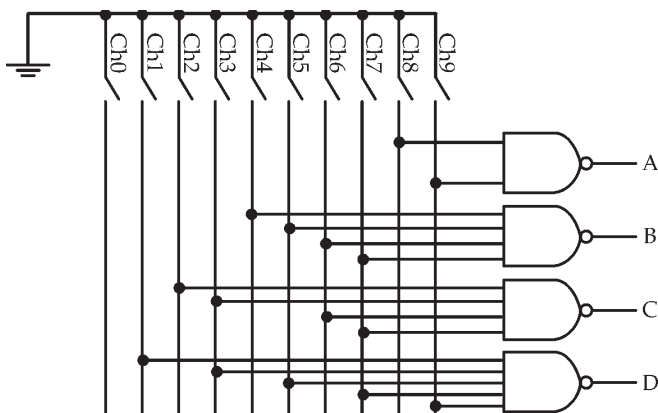
A = 1, quando: Ch8 ou Ch9 for acionada.

B = 1, quando: Ch4, Ch5, Ch6, ou Ch7 for acionada.

C = 1, quando: Ch2, Ch3, Ch6, ou Ch7 for acionada.

D = 1, quando: Ch1, Ch3, Ch5, Ch7 ou Ch9 for acionada.

Desta forma, o circuito lógico é dado por:



5.2.2 Decodificador Binário-Decimal

Será montada a tabela da verdade do circuito cujas entradas são bits do código BCD 8421 e as saídas são os respectivos bits do código decimal 9876543210.

BCD 8421				Código 9876543210									
A	B	C	D	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0
0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	1	0	0	0	0
0	1	0	1	0	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	0	1	0	0	0	0	0	0
0	1	1	1	0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0	0	0	0	0	0	0

O próximo passo é transpor cada saída da tabela para o diagrama de Karnaugh. Deve-se observar que o código BCD 8421 não possui números maiores que 9 e assim, tanto faz o valor assumido nas possibilidades excedentes, o que indica que estes valores são irrelevantes (x) no mapa de Karnaugh.

	\bar{C}	C		
\bar{A}	0	0	0	0
	0	0	0	0
A	x	x	x	x
	0	1	x	x
	\bar{D}	D	\bar{D}	

Mapa para S9

$$S9 = AD$$

	\bar{C}	C		
\bar{A}	0	0	0	0
	0	0	0	0
A	x	x	x	x
	1	0	x	x
	\bar{D}	D	\bar{D}	

Mapa para S8

$$S8 = A\bar{D}$$

	\bar{C}	C		
\bar{A}	0	0	0	0
	0	0	1	0
A	x	x	x	x
	0	0	x	x
	\bar{D}	D	\bar{D}	

Mapa para S7

$$S7 = BCD$$

	\bar{C}	C		
\bar{A}	0	0	0	0
	0	0	0	1
A	x	x	x	x
	0	0	x	x
	\bar{D}	D	\bar{D}	

Mapa para S6

$$S6 = B\bar{C}\bar{D}$$

	\overline{C}		C		
\overline{A}	0	0	0	0	\overline{B}
A	0	1	0	0	B
	X	X	X	X	
	0	0	X	X	\overline{B}
	\overline{D}	D	\overline{D}		

Mapa para S5

$$S5 = B\overline{C}D$$

	\overline{C}		C		
\overline{A}	0	0	0	0	\overline{B}
A	1	0	0	0	B
	X	X	X	X	
	0	0	X	X	\overline{B}
	\overline{D}	D	\overline{D}		

Mapa para S4

$$S4 = B\overline{C}\overline{D}$$

	\overline{C}		C		
\overline{A}	0	0	0	0	\overline{B}
A	0	0	0	0	B
	X	X	X	X	
	0	1	X	X	\overline{B}
	\overline{D}	D	\overline{D}		

Mapa para S9

$$S9 = AD$$

	\overline{C}		C		
\overline{A}	0	0	0	0	\overline{B}
A	0	0	0	0	B
	X	X	X	X	
	1	0	X	X	\overline{B}
	\overline{D}	D	\overline{D}		

Mapa para S8

$$S8 = A\overline{D}$$

5.2.3 Decodificador BCD 8421 → Excesso 3

Será projetado o circuito que decodifica o código BCD 8421 para excesso 3.

BCD 8421				Excesso 3			
A	B	C	D	S3	S2	S1	S0
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

Para simplificar as expressões, monta-se o diagrama de Veitch-Karnaugh.

	\bar{C}	C		
	0	0	0	0
\bar{A}	0	1	1	1
A	X	X	X	X
	1	1	X	X
	\bar{D}	D	\bar{D}	

Mapa para S3

$$S3 = A + BD + BC$$

	\bar{C}	C		
	0	1	1	1
\bar{A}	1	0	0	0
A	X	X	X	X
	0	1	X	X
	\bar{D}	D	\bar{D}	

Mapa para S2

$$S2 = \bar{B}D + \bar{B}\bar{D} + B\bar{C}\bar{D}$$

	\bar{C}	C		
	1	0	1	0
\bar{A}	1	0	1	0
A	X	X	X	X
	1	0	X	X
	\bar{D}	D	\bar{D}	

Mapa para S1

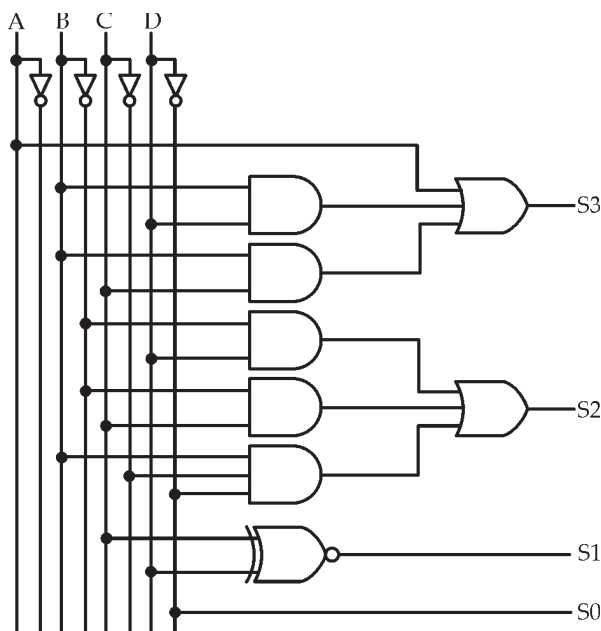
$$S1 = \bar{C}\bar{D} + CD + C\odot D$$

	\bar{C}	C		
	1	0	0	1
\bar{A}	1	0	0	1
A	X	X	X	X
	X	0	X	X
	\bar{D}	D	\bar{D}	

Mapa para S0

$$S0 = \bar{D}$$

O circuito decodificador, obtido das expressões simplificadas é dado por:

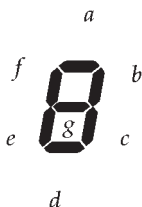


Seguindo os procedimentos adotados é possível construir qualquer circuito codificador/decodificador que possibilita a conversão entre qualquer código.

5.2.4 Decodificador para Display de 7 segmentos

O display de 7 segmentos possibilita a escrita de números decimais de 0 a 9 e alguns outros símbolos que podem ser letras ou sinais. A Figura a seguir ilustra um display genérico com a nomenclatura de identificação dos segmentos.

Existem várias tecnologias de fabricação de display e será utilizada a mais comum, que é o display a led. Existem dois tipos: catodo comum e anodo comum.



Existem várias tecnologias de fabricação de display e será utilizada a mais comum, que é o display a led. Existem dois tipos: catodo comum e anodo comum.






O catodo comum possui todos os catodos dos led's interligados e, desta forma, necessita-se aplicar nível 1 em cada anodo para acender. No display tipo anodo comum é necessário aplicar nível 0 ao catodo correspondente para acender.

A título de exemplo será elaborado um decodificador, a partir de um código BCD 8421, que escreve a seqüência de 0 a 9 em um display de 7 segmentos de catodo comum (aplica-se nível 1 para acender).

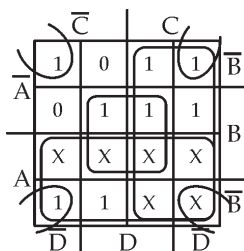


A tabela abaixo mostra o código de entrada de 4 bits e os níveis aplicados em cada segmento.

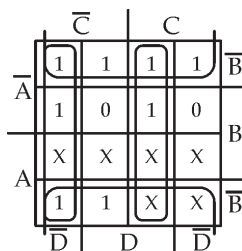
Caracteres	Display	BCD 8421				Código par 7 segmentos						
		A	B	C	D	a	b	c	d	e	f	g
0		0	0	0	0	1	1	1	1	1	1	0
1		0	0	0	1	0	1	1	0	0	0	0
2		0	0	1	0	1	1	0	1	1	0	1
3		0	0	1	1	1	1	1	1	0	0	1
4		0	1	0	0	0	1	1	0	0	1	1

5		0 1 0 1	1 0 1 1 0 1 1
6		0 1 1 0	1 0 1 1 1 1 1
7		0 1 1 1	1 1 1 0 0 0 0
8		1 0 0 0	1 1 1 1 1 1 1
9		1 0 0 1	1 1 1 1 0 1 1

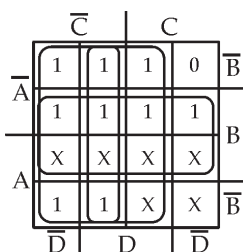
Para simplificar o circuito de saída basta utilizar o diagrama de Karnaugh. Os termos que não são representados na tabela serão considerados irrelevantes.



Mapa para (a)
 $a = A + C + B \odot D$

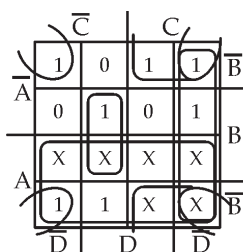


Mapa para (b)
 $b = \overline{B} + C \odot D$



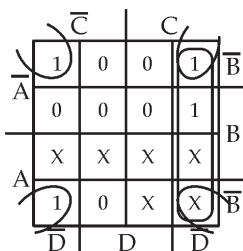
Mapa para (c)

$$C = B + \overline{C} + D$$



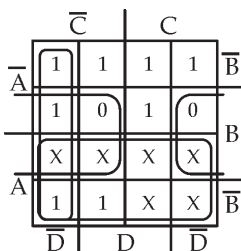
Mapa para (d)

$$d = A + \overline{B}\overline{D} + \overline{B}C + C\overline{D} + B\overline{C}$$



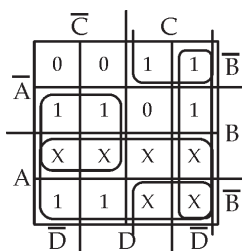
Mapa para (e)

$$e = \overline{B}\overline{D} + \overline{C}\overline{D}$$



Mapa para (f)

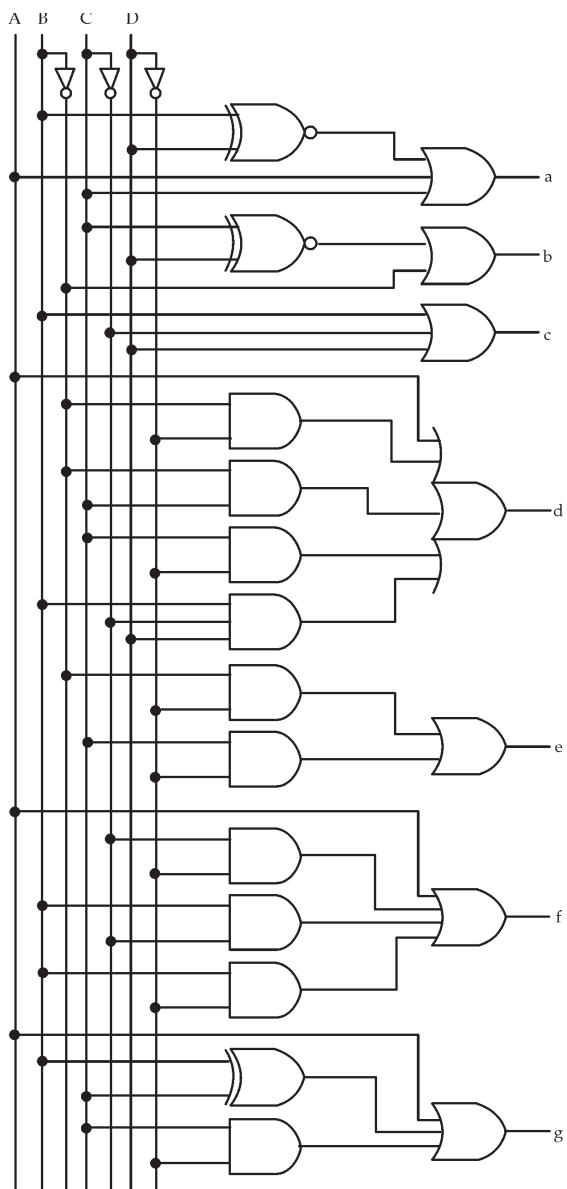
$$f = A + \overline{B}\overline{D} + \overline{B}\overline{C} + B\overline{D}$$



Mapa para (g)

$$g = A + C\overline{D} + B \oplus C$$

O circuito lógico obtido das expressões simplificadas é visto na figura a seguir.



EXERCÍCIOS PROPOSTOS:

- 1) Elabore um codificador Decimal/Binário para, a partir de um teclado com chaves numeradas de 0 a 3, fornecer nas saídas o código correspondente. Considere que as entradas das portas em vazio equivalem à aplicação de nível lógico 1.
- 2) Projete um circuito combinacional para em um conjunto de 4 fios, fornecer nível 0 em apenas um deles por vez (estando os demais em nível 1), conforme seleção binária aplicada às entradas digitais.
- 3) Elabore um decodificador 3 para 8 onde, conforme as combinações entre os 3 fios de entrada, 1 entre os 8 fios de saída é ativado (nível 1).
- 4) Desenvolva um circuito que transforme o código BCD8421 para o código de Johnson.
- 5) Projete um decodificador para, a partir de um código binário, escrever a sequência de 1 a 5 em um display de 7 segmentos catodo comum.

CONSULTAS RECOMENDADAS:

Livro Texto: Sistemas Digitais: Princípios e Aplicações, 8ª
Edição - Ronald J. Tocci e Neal S. Widmer – São Paulo:
Prentice Hall, 2003 - Capítulo 9.

MÓDULO 6

Aritmética Digital: Operações e Circuitos

RESUMO

Para entender como os computadores e as calculadoras digitais realizam as várias operações aritméticas sobre números representados no formato binário, faz-se necessário estudar essas operações usando “lápiz e papel”. Apesar de os vários circuitos aritméticos serem disponibilizados em CI's, para entender o funcionamento desses é importante saber projeta-los.

OBJETIVO

Este módulo aborda os princípios básicos necessários para entender como as máquinas digitais realizam as operações aritméticas básicas, bem como o projeto de circuitos somadores e subtratores utilizados nos equipamentos digitais.

CONTEÚDO

6.1 Operações Aritméticas no Sistema Binário

Nas áreas de Eletrônica Digital e dos Microprocessadores, o estudo das operações aritméticas no sistema binário é muito importante, pois estas serão utilizadas em circuitos aritméticos, que serão estudados posteriormente.

6.1.1 Adição no Sistema Binário

A adição no sistema binário é efetuada de maneira idêntica ao sistema decimal, entretanto, apenas quatro casos podem ocorrer na soma de dois dígitos binários:

$$0+0=0$$

$$0+1=1$$

$$1+1=10=0 + \text{vai 1 para a próxima posição}$$

$$1+1+1=11=1+\text{vai 1 para próxima posição}$$

Para exemplificar serão realizadas as seguintes adições:

$$\begin{array}{r}
 1 \leftarrow \\
 11 \\
 +10 \\
 \hline
 101
 \end{array}
 \begin{array}{l}
 \text{Transporte}
 \end{array}$$

$$\begin{array}{r}
 11 \leftarrow \\
 110 \\
 +111 \\
 \hline
 1101
 \end{array}
 \begin{array}{l}
 \text{Transporte}
 \end{array}$$

Nota-se, então que a adição é realizada coluna a coluna, considerando sempre o transporte proveniente da coluna anterior.

6.1.2 Subtração no Sistema Binário

O método de subtração é análogo a uma subtração no sistema decimal. Assim, tem-se:

$$\begin{array}{r}
 0 \\
 -0 \\
 \hline
 0
 \end{array}
 \begin{array}{r}
 0 \\
 -1 \\
 \hline
 1
 \end{array}
 \begin{array}{r}
 1 \\
 -0 \\
 \hline
 1
 \end{array}
 \begin{array}{r}
 1 \\
 -1 \\
 \hline
 0
 \end{array}$$

Para o caso 0 -1, o resultado será igual a 1, porém haverá um transporte para a coluna seguinte que deve ser acumulado no subtraendo e, obviamente, subtraído do minuendo. Para exemplificar, tem-se:

$$\begin{array}{r}
 111 \\
 -100 \\
 \hline
 011
 \end{array}
 \begin{array}{r}
 1011 \\
 1 \leftarrow \\
 -101 \\
 \hline
 0110
 \end{array}
 \begin{array}{l}
 \text{Transporte}
 \end{array}$$

6.1.3 Multiplicação no Sistema Binário

Ocorre exatamente como uma multiplicação no sistema decimal. Assim sendo, tem-se:

$$\begin{array}{l}
 0 \times 0 = 0 \\
 0 \times 1 = 0 \\
 1 \times 0 = 0 \\
 1 \times 1 = 1
 \end{array}$$

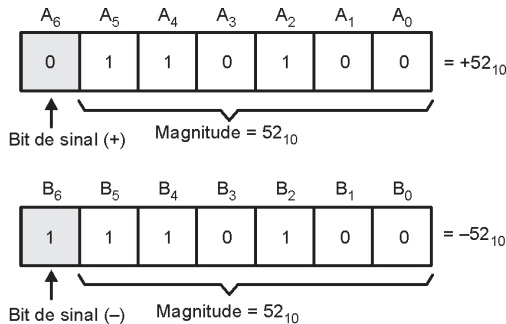
Para exemplificar, efetua-se a multiplicação entre os números 110102 e 1012.

$$\begin{array}{r}
 11010 \\
 \times 101 \\
 \hline
 11010 \\
 00000+ \\
 11010++ \\
 \hline
 10000010
 \end{array}$$

6.1.4 Notação de Números Binários Positivos e Negativos

A representação de números binários positivos e negativos pode ser feita utilizando-se os sinais + ou – respectivamente. Na prática, estes sinais não podem ser utilizados, pois tudo deve ser codificado em 0 ou 1, nos hardwares que processam as operações aritméticas. Desta forma, utiliza-se um **bit de sinal** colocado na posição de algarismo mais significativo. Se o número for positivo, o bit de sinal será 0, se o número for negativo este será 1. Este processo de representação é conhecido por **sinal-módulo**.

Exemplo de representação de números com sinal usando sinal-módulo:



Uma outra forma de representar número binário negativo é a notação do **complemento de 2**. Para obtê-lo é necessário primeiramente converter o número em complemento de 1.

Forma do complemento de 1:

A obtenção do complemento de 1 de um número binário se dá pela troca de cada bit do número pelo seu inversor ou complemento, ou seja, o complemento de 1 de 10011011₂ é 01100100₂.

Forma do complemento de 2:

O complemento de 2 é obtido somando-se 1 ao complemento de 1 do número binário inicial.

Exemplo: 11001101_2

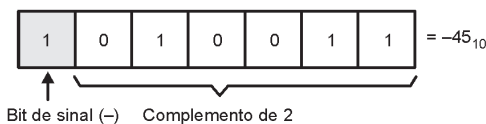
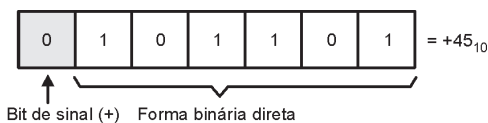
Complemento de 1: 00110010

Complemento de 2: $\begin{array}{r} 00110010 \\ +1 \\ \hline 00110011 \end{array}$

Exemplo de representação de números com sinal usando complemento de 2:

- **Se o número for positivo:** magnitude é representada na forma binária direta, e um bit de sinal 0 é colocado em frente ao bit MSB.

- **Se o número for Negativo:** magnitude é representada na sua forma de complemento de 2 e um bit de sinal 1 é colocado em frente ao bit MSB.



Exercício: represente cada um dos números decimais com sinal como um número binário com sinal no sistema de complemento de 2. Use um total de 5 bits, incluindo o bit de sinal.

a) +13; b) -2; c) -8

6.1.5 Adição no Sistema de Complemento de 2

a) dois números positivos: +9 com +4



- b) Um número positivo e um outro menor e negativo

$$\begin{array}{r}
 +9 \rightarrow \boxed{0} \boxed{1} \boxed{0} \boxed{0} \boxed{1} \\
 -4 \rightarrow \boxed{1} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \\
 \hline
 \nearrow \boxed{0} \boxed{0} \boxed{1} \boxed{0} \boxed{1} \quad (\text{Soma} = 00101 = +5)
 \end{array}$$

\downarrow bits de sinal
 esse vai um é desconsiderado

- c) um número positivo e outro maior e negativo.

$$\begin{array}{r}
 -9 \rightarrow \boxed{1} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \\
 -4 \rightarrow \boxed{0} \boxed{0} \boxed{1} \boxed{0} \boxed{0} \\
 \hline
 \downarrow \boxed{1} \boxed{1} \boxed{0} \boxed{1} \boxed{1} \quad (\text{Soma} = 11011 = -5)
 \end{array}$$

\downarrow bits de sinal

- d) Dois números negativos

$$\begin{array}{r}
 -9 \rightarrow \boxed{1} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \\
 -4 \rightarrow \boxed{1} \boxed{1} \boxed{1} \boxed{0} \boxed{0} \\
 \hline
 \nearrow \boxed{1} \boxed{0} \boxed{0} \boxed{1} \boxed{1} \quad (\text{Soma} = 10011 = -13)
 \end{array}$$

\downarrow bits de sinal
 esse vai um é desconsiderado

- e) Dois números iguais e de sinais opostos

$$\begin{array}{r}
 -9 \rightarrow \boxed{1} \boxed{0} \boxed{1} \boxed{1} \boxed{1} \\
 +9 \rightarrow \boxed{0} \boxed{1} \boxed{0} \boxed{0} \boxed{1} \\
 \hline
 \downarrow \boxed{1} \boxed{0} \boxed{0} \boxed{1} \boxed{0} \quad (\text{Soma} = 0000)
 \end{array}$$

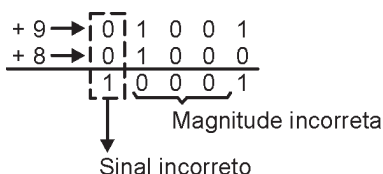
\downarrow esse vai um é desconsiderado

6.1.6 Subtração no Sistema de Complemento de 2

A subtração no sistema de complemento de 2 envolve a adição:

- Faça a operação de Negação do subtraendo através do complemento de 2.
- Adicione esse número obtido ao minuendo.

6.1.7 Overflow Aritmético



6.1.8 Multiplicação no Sistema de Complemento de 2

Quando os dois números são positivos, eles já estarão no formato binário direto e poderão ser multiplicados nesse formato. O resultado será também um número positivo tendo um bit de sinal de 0.

Quando os dois números são negativos, eles deverão estar na forma de complemento de 2. Deve-se aplicar o complemento de 2 para torna-los positivo e efetuar a multiplicação que terá como resultado um número positivo e o bit de sinal será 0.

Quando um número for positivo e o outro negativo, o número negativo é convertido para a forma positiva. Faz-se a multiplicação e aplica-se o complemento de 2 ao resultado para torna-lo negativo.

6.2. Circuitos Aritméticos

Circuitos aritméticos são circuitos combinacionais utilizados, principalmente, para construir a ULA (Unidade Lógica Aritmética) dos microprocessadores e são encontrados disponíveis em circuitos integrados comerciais.

6.2.1. Meio Somador

O meio somador possibilita efetuar a soma de números binários com somente 1 algarismo.

Assim, pode-se construir a tabela da verdade da soma de 2 números binários de 1 algarismo, definindo T_S como transporte de saída.

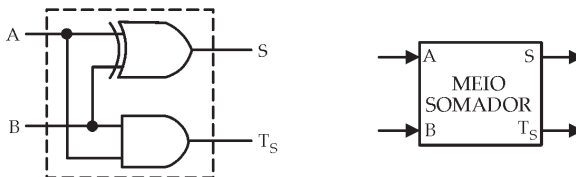
A	B	S	T _S
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Utilizando a tabela, pode-se montar um circuito que possui como entrada as variáveis booleanas **A** e **B**, e como saída, a soma dos algarismos **S** e o respectivo transporte de saída **T_S**. As expressões características extraídas da tabela são:

$$S = A \oplus B \quad \text{e} \quad T_S = AB$$

Circuito extraído das equações acima.

Representação em blocos do circuito.



O meio somador é conhecido por *Half adder* e o transporte T_S por *carry out*.

6.2.2 Somador Completo

O somador completo é um circuito lógico utilizado para fazer a soma de 2 números binários de mais de 1 algarismo, pois possibilita a introdução do transporte de entrada C_{IN} proveniente da coluna anterior.

A tabela da verdade do somador completo está descrita abaixo.

A	B	T _E	S	T _S
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

As expressões características, sem simplificações, de um somador completo são:

$$S = \bar{A}\bar{B}T_E + \bar{A}B\bar{T}_E + A\bar{B}\bar{T}_E + ABT_E$$

$$T_S = \bar{A}BT_E + A\bar{B}T_E + AB\bar{T}_E + ABT_E$$

Simplificando com o Mapa de Karnaugh:

	\bar{T}_E	T_E
$\bar{A}\bar{B}$	0	1
$\bar{A}B$	1	0
AB	0	1
$A\bar{B}$	1	0

$$S = A \oplus B \oplus T_E$$

	\bar{T}_E	T_E
$\bar{A}\bar{B}$	0	0
$\bar{A}B$	0	1
AB	1	1
$A\bar{B}$	0	1

$$T_S = AT_E + BT_E + AB$$

Das equações simplificadas é montado o circuito lógico do somador completo.

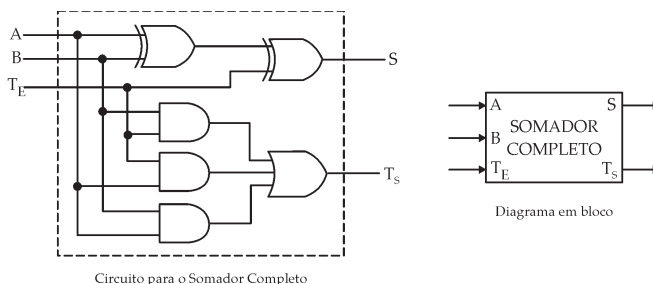


Figura 2 – Circuito e diagrama em blocos do Somador Completo

O circuito somador completo é conhecido por **Full Adder**, sendo a entrada do transporte T_E denominada de **carry in**.

Para exemplificar, será montado um sistema em blocos que efetua a soma de dois números de 5 bits, conforme o esquema a seguir. Este raciocínio pode ser estendido para qualquer quantidade de bits:

Seja: $A = A_4A_3A_2A_1A_0$ e $B = B_4B_3B_2B_1B_0$

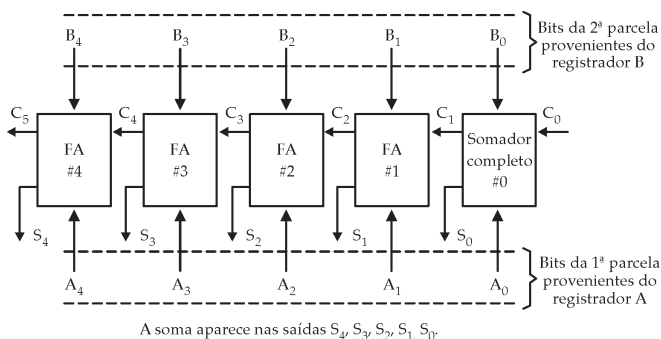


Figura 3 – Diagrama em bloco de um circuito Somador Paralelo Usando Somadores Completos

Obs. Para se efetuar a soma dos bits A_0 e B_0 pode-se utilizar um meio somador, pois não existe transporte de entrada. Para as demais colunas deve-se utilizar o somador completo, pois **TE (carry in)** deve ser considerado.

6.2.4 Propagação do Carry

No circuito da figura 3, o bit S_4 do último somador completo depende do bit C_1 do primeiro somador completo. Porém C_1 tem de passar pelos quatro FAs antes de gerar a saída S_4 . Isso representa um atraso de tempo que depende do atraso de cada somador completo. Supondo que cada FA tenha um atraso de 40ns, S_4 não alcançará o resultado correto até que tenha transcorrido 200ns. Quanto maior o número de bits maior o atraso.

Para reduzir esse atraso pode-se usar um circuito de geração de carry antecipado.

6.2.5 Meio Subtrator

O meio subtrator efetuar a subtração de 2 números binários com somente 1 algarismo. Desta forma, pode-se montar a tabela da verdade considerando a operação de subtração de 2 números binários de 1 algarismo ($A - B$).

A	B	S	T_s
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Pode-se montar o circuito lógico que executa a tabela, tendo como entrada as variáveis booleanas **A** e **B**, e como saída, a subtração **S** e o transporte de saída **TS**. As expressões características extraídas do circuito são:

$$S = A \oplus B \text{ e } T_s = \bar{A}B$$

Circuito extraído das equações acima

Representação em blocos do circuito

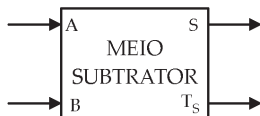
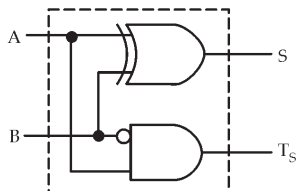


Figura 4 – Circuito e diagrama em blocos meio subtrator

O meio subtrator é conhecido por **Half subtractor**.

6.2.6 Subtrator Completo

O subtrator completo é utilizado para fazer a subtração de 2 números binários de mais de 1 algarismo, pois possibilita a introdução do transporte de entrada T_E proveniente da coluna anterior.

A tabela da verdade do subtrator completo é dada por:

A	B	T_E	S	T_S
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

As expressões características, sem simplificações, de um subtrator completo são:

$$S = \bar{A}\bar{B}T_E + \bar{A}B\bar{T}_E + A\bar{B}\bar{T}_E + ABT_E$$

$$T_S = \bar{A}\bar{B}T_E + \bar{A}B\bar{T}_E + \bar{A}BT_E + ABT_E$$

Simplificando com o Mapa de Karnaugh:

	\bar{T}_E	T_E	
$\bar{A}\bar{B}$	0	1	$S = A \oplus B \oplus T_E$
$\bar{A}B$	1	0	
AB	0	1	
$A\bar{B}$	1	0	

	\bar{T}_E	T_E	
$\bar{A}\bar{B}$	0	1	$T_S = \bar{A}T_E + \bar{A}B + BT_E$
$\bar{A}B$	1	1	
AB	0	1	
$A\bar{B}$	0	0	

Das equações simplificadas é montado o circuito lógico do somador completo.

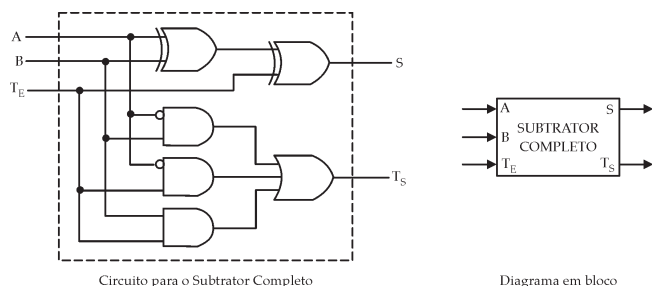


Figura 5 – Circuito e diagrama em blocos do SubtratorCompleto

O subtrator completo é conhecido por **Full subtractor**.

Da mesma forma, pode-se esquematizar um sistema subtrator para 2 números de m bits, onde $m = n + 1$.

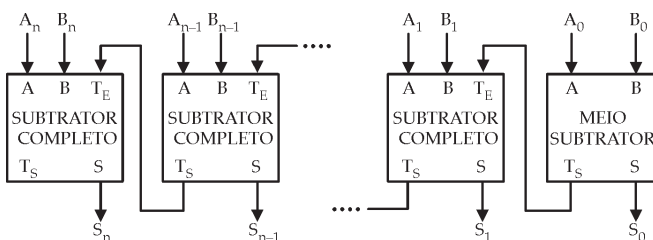


Figura 6 – Diagrama em bloco de um circuito Subtrator Paralelo Usando Subtratores Completos

Neste sistema, a saída de transporte TS do último bloco é desnecessária se o minuendo ($A_n \dots A_0$) for maior ou igual ao subtraendo ($B_n \dots B_0$), porém poderá ser utilizada no caso contrário para indicar que o resultado é negativo, estando, então, na notação do complemento de 2.

6.2.6 Sistema de Complemento de 2

Quando é usado complemento de 2 para representar números negativos, as operações de adição e subtração podem ser realizadas usando apenas a adição.

A figura 7 mostra um somador paralelo usado para somar um número positivo com um número negativo na forma de complemento de 2, no caso -3 com +6.

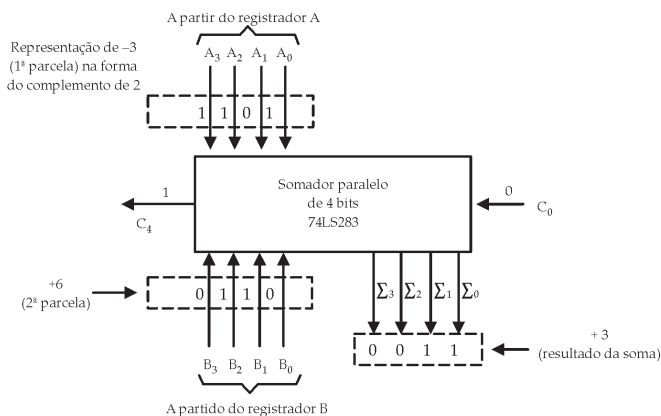


Figura 7 – somador paralelo usado para somar um número positivo com um número negativo na forma de completo de 2. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

Conforme abordado anteriormente, quando o sistema de complemento de 2 é usado, o número a ser subtraído (subtraendo) é transformado para a sua forma de complemento de 2 e então somado ao minuendo. A figura 8 mostra um somador paralelo usado para realizar uma subtração no sistema de complemento de 2.

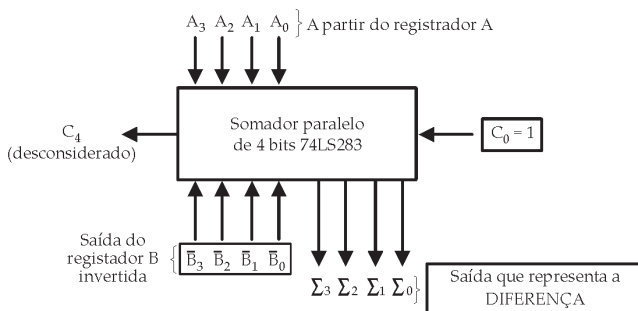


Figura 8 – Somador paralelo usado para realizar uma subtração ($A - B$) usando o sistema do complemento de 2. Os bits do subtraendo (B) são invertidos e $C_0 = 1$ para gerar o complemento de 2. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

EXERCÍCIOS PROPOSTOS:

Resolva os problemas 6.18 a 6.20, 6.23 e 6.25 do capítulo 6 do **Livro Texto**.

CONSULTAS RECOMENDADAS:

Livro Texto: Sistemas Digitais: Princípios e Aplicações, 8ª
Edição - Ronald J. Tocci e Neal S. Widmer – São Paulo:
Prentice Hall, 2003 - Capítulo 6.

MÓDULO 7

Flip-Flop, Registradores e Contadores

RESUMO

A maioria dos circuitos dos sistemas digitais é constituída de circuitos combinacionais e de elementos de memória. O elemento de memória mais importante é o flip-flop que é implementado a partir de portas lógicas. Diferentes tipos de flip-flops podem ser combinados e podem ser usados em uma ampla variedade de aplicações incluindo contagem, armazenamento binário de dados e transferência de dados.

OBJETIVO

Construir e analisar o funcionamento de flip-flops com portas NAND e NOR, estudar as diferenças entre sistemas síncronos e assíncronos, entender o funcionamento dos flip-flops disparados por transição e suas aplicações.

CONTEÚDO

7.1 Introdução

Será iniciado o estudo dos circuitos seqüenciais que, por definição, são os circuitos que apresentam as saídas dependentes das variáveis de entrada e/ou de seus estados anteriores que permanecem armazenados, operando sob o comando de uma seqüência de pulsos denominados de **clock**.

7.1 Flip-Flop

É o elemento de memória mais importante. É implementado a partir de portas lógicas. A Fig. Mostra um tipo de símbolo genérico usado para representar um flip-flop (FF). Esse símbolo apresenta duas saídas, denominadas Q (saída normal) e \bar{Q} (saída invertida), opostas entre si.

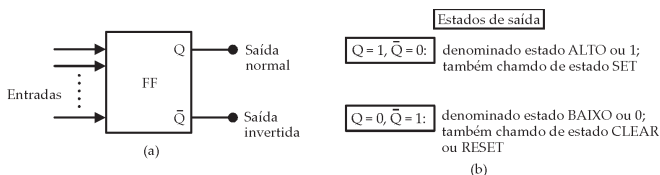


Figura 1- Símbolo geral para um flip-flop e definição dos seus dois estados de saída possíveis. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

7.1.1 Latch com Portas NAND

O circuito mais simples de um FF pode ser construído a partir de duas portas NAND ou duas portas NOR. A figura 2 mostra um latch com portas NAND e sua tabela-verdade.

Resumo do funcionamento:

- SET = CLEAR = 1: estado de repouso, não tem nenhum efeito sobre o estado de saída; Q e \bar{Q} permanecem inalteradas.
- SET = 0, CLEAR = 1: faz a saída ir para o estado ALTO, $Q=1$, operação de SETAR o latch.
- SET = 1, CLEAR = 0: faz a saída ir para o estado BAIXO, $Q=0$, operação de RESETAR ou limpar o latch.
- SET = 0, CLEAR = 0: esta condição tenta ao mesmo tempo SETAR e RESETAR o latch e não deve ser usada.

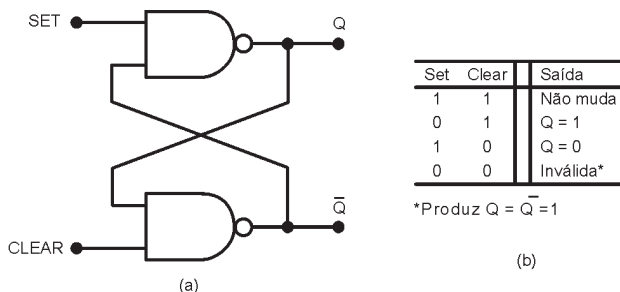


Figura 2- (a) Latch NAND; (b) Tabela-verdade. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

Em um latch NAND as entradas SET e CLEAR são ativadas em nível BAIXO, por isso, o latch com portas NAND pode ser representado usando a representação equivalente para portas NAND mostrada na figura 3. Os pequenos círculos nas entradas, assim como a os nomes dos sinais $\overline{\text{SET}}$ e $\overline{\text{CLEAR}}$ indicam o estado de ativação em nível BAIXO dessas entradas.

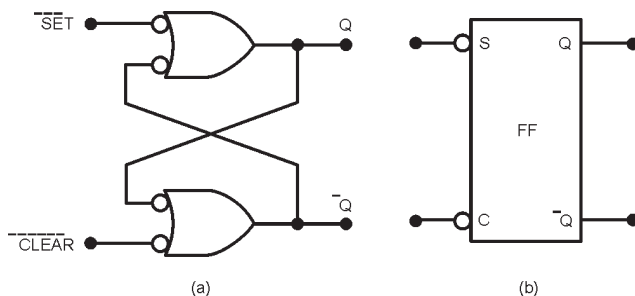


Figura 3 Representação equivalente de um latch NAND; (b) símbolo simplificado. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

Exemplo1: Dadas as formas de onda de entrada, determine a forma de onda na saída Q, considerando inicialmente Q=0.

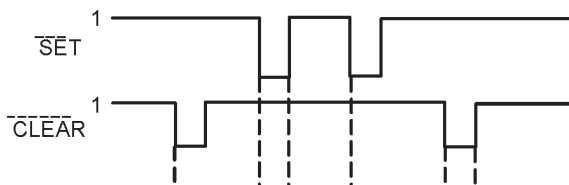


Figura 4 – Exemplo 1. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

Exemplo 2: A trepidação de um contato mecânico gera múltiplas transições na tensão, um latch NAND pode ser usado para eliminar as múltiplas transições na tensão. Descreva o funcionamento do circuito da figura 5(b) que elimina o efeito de trepidação.

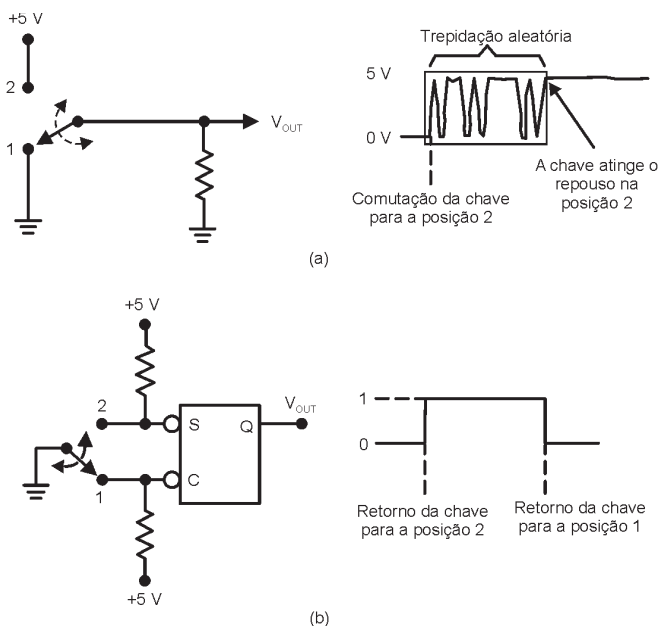


Figura 5 – Exemplo 2. Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer

7.1.2 Latch com Portas NOR

A figura 6 mostra um latch com portas NOR e sua tabela-verdade. Observe que as saídas Q e \bar{Q} estão trocadas em relação ao latch NAND.

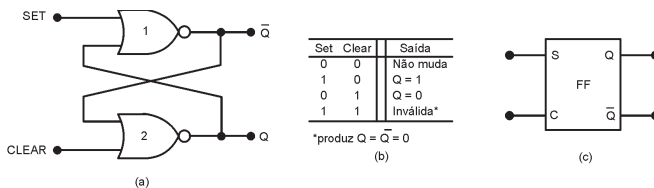


Figura 6- (a) Latch NOR; (b) Tabela-verdade; (c) Símbolo simplificado. Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer

Resumo do funcionamento:

- a) SET = CLEAR = 0: estado de repouso, não tem nenhum efeito sobre o estado de saída; Q e \bar{Q} permanecem inalteradas.
- b) SET = 1, CLEAR = 0: faz a saída ir para o estado ALTO, Q=1, operação de SETAR o latch.
- c) SET = 0, CLEAR = 1: faz a saída ir para o estado BAIXO, Q=0, operação de RESETAR ou limpar o latch.
- d) SET = 1, CLEAR = 1: esta condição tenta ao mesmo tempo SETAR e RESETAR o latch e gera Q = \bar{Q} = 0.

Exemplo 3: Considere que inicialmente Q=0 e determine a forma de onda na saída Q para um latch NOR que tem as entradas mostradas na figura abaixo.

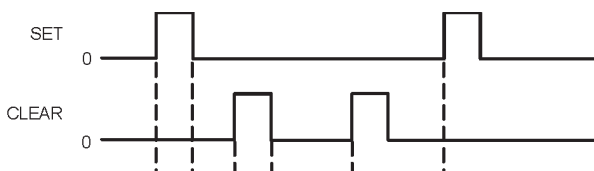


Figura 7 – Exemplo 3. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

Exemplo 4: A figura abaixo mostra um circuito simples que pode ser usado para detectar a interrupção de um feixe de luz. A luz é focalizada em um fototransistor em emissor-comum para operar como uma chave. Considere que o latch tenha sido previamente levado para o estado 0 ao abrir a chave SW1 momentaneamente e descreva o que acontece se o feixe de luz for momentaneamente interrompido.

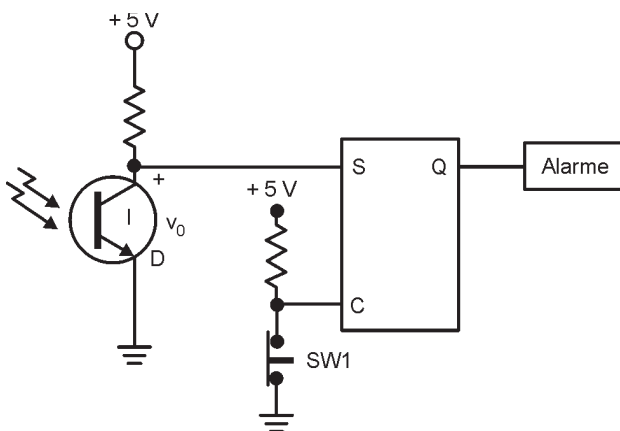


Figura 8 – Exemplo 4. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

7.1.3 Sinais de Clock e Flip-Flops com Clock

Nos sistemas assíncronos, as saídas de circuitos lógicos podem mudar de estado a qualquer momento em que uma ou mais entradas mudarem de estado.

Em sistemas síncronos, os momentos exatos em que uma saída qualquer pode mudar de estado são determinados por um sinal normalmente determinado clock. A figura 9 mostra exemplos de sinais de clock.

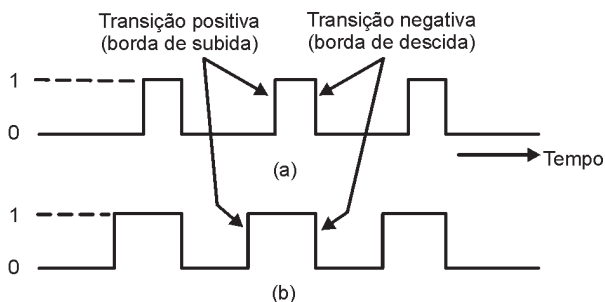


Figura 9 – Sinais de clock. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

Flip-flop com clock tem uma entrada de clock (CLK) que pode ser ativada por uma borda de subida, figura 10 (a), ou por uma borda de descida, figura 10 (b). As entradas de controle determinam o efeito da transição ativa do clock.

As entradas de controle determinam o que ocorrerá com as saídas; a entrada CLK determina quando as saídas serão alteradas em função das entradas.

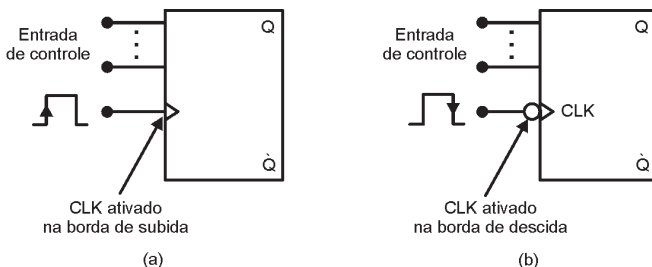


Figura 10 – Símbolos de Flip-flops com clock: (a) disparado na subida; (b) disparado na descida. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

Tempo de setup t_s (preparação): intervalo de tempo durante o qual as entradas têm de ser mantidas no nível adequado, imediatamente **antes** da transição ativa do clock, figura 11(a).

Tempo de hold t_h (manutenção): intervalo de tempo durante o qual as entradas têm de ser mantidas no nível adequado, imediatamente **após** da transição ativa do clock, figura 11(b).

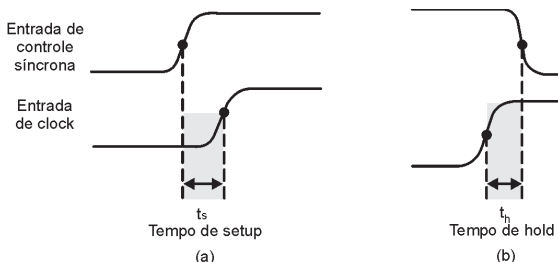


Figura 11 – Entradas de controle tem de ser mantidas estáveis por (a) um tempo t_s antes da transição ativa do clock e por (b) um tempo t_h após a transição ativa do clock. *Sistemas Digitais: Princípios e Aplicações* - Ronald J. Tocci e Neal S. Widmer

7.1.4. Flip-Flop SC com clock

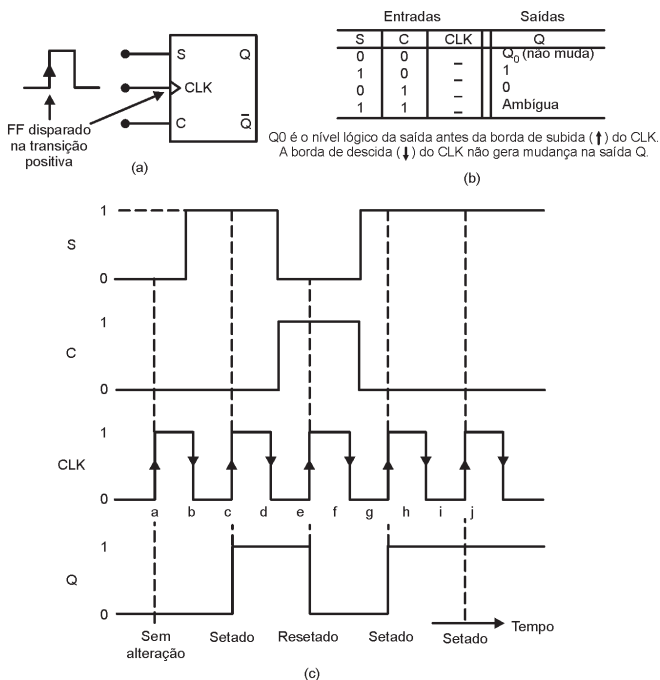


Figura 12 – (a) Flip-flop SC com clock que responde apenas à borda positiva do pulso de clock; (b) tabela-verdade; (c) formas de onda típicas. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

A figura 13 mostra a versão simplificada do circuito interno de um flip-flop SC disparado por borda.

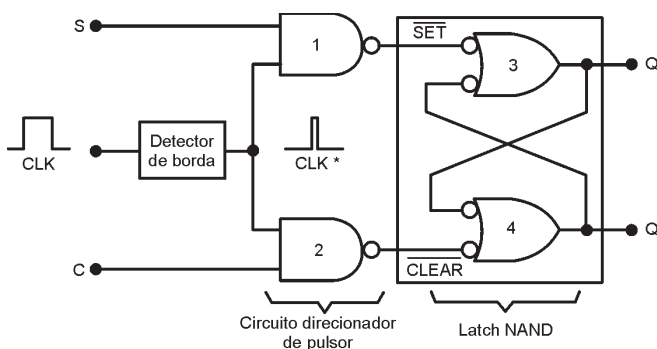


Figura 13 – circuito interno de um flip-flop s-c disparado por borda. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

7.1.5. Flip-Flop J-K com clock

As entradas de controle J e K controlam o estado lógico do FF: a condição $J = K = 1$ não resulta em uma saída ambígua. Para essa condição o FF comuta para o estado lógico oposto ao estado presente no instante da transição positiva do clock, figura 14.

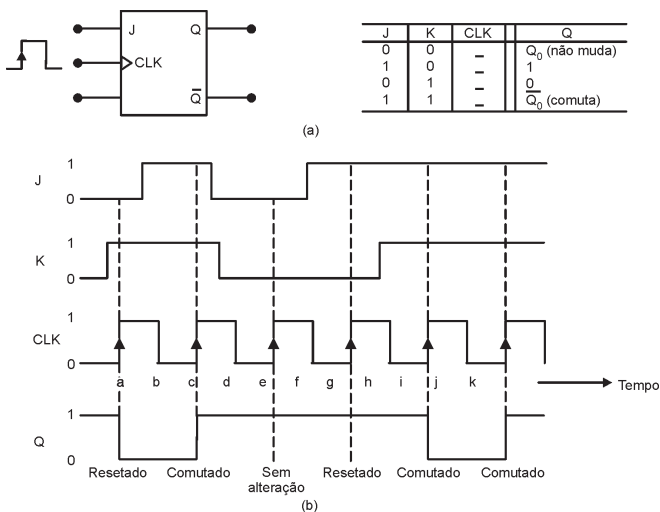


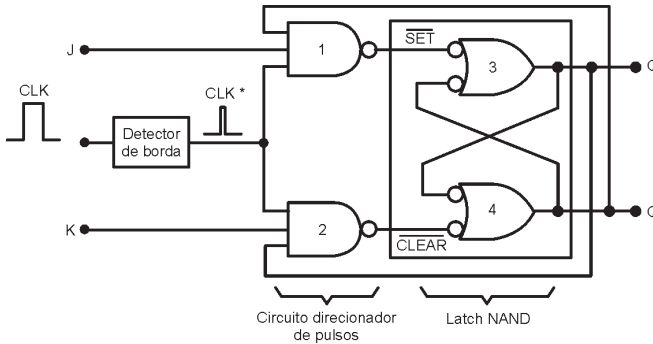
Figura 14 – Flip-flop J-K disparado na borda positiva. *Sistemas Digitais: Princípios e Aplicações - Ronald J. Tocci e Neal S. Widmer*

Circuito interno FLIP-FLOP J-K disparado por borda:

A figura 15 mostra o circuito interno de um Flip-flop J-K:

$J=K=1; Q=0 \Rightarrow$ A porta NAND 1 direciona CLK* (invertido) para a entrada SET do latch NAND gerando $Q=1$;

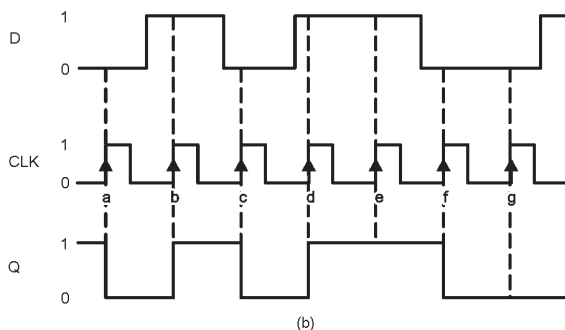
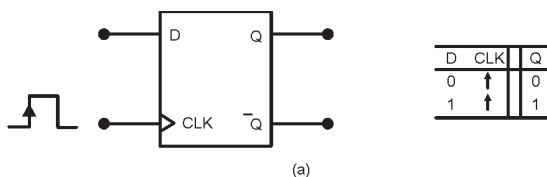
$J=K=1; Q=1 \Rightarrow$ A porta NAND 2 direciona CLK* (invertido) para a entrada CLEAR do latch NAND gerando $Q=0$



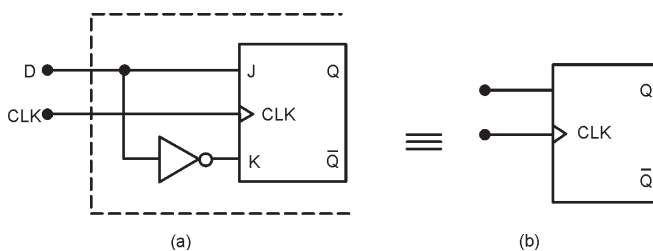
7.1.6. Flip-Flop D com clock

A saída Q irá para o mesmo estado lógico presente na entrada D quando ocorrer uma borda de subida do clock.

A figura 16 mostra um flip-flop D disparados apenas nas transições positivas do clock (a) e as Formas de onda típicas (b). O nível lógico presente na entrada D é armazenado no Flip-flop no instante em que ocorre a borda de subida do clock.



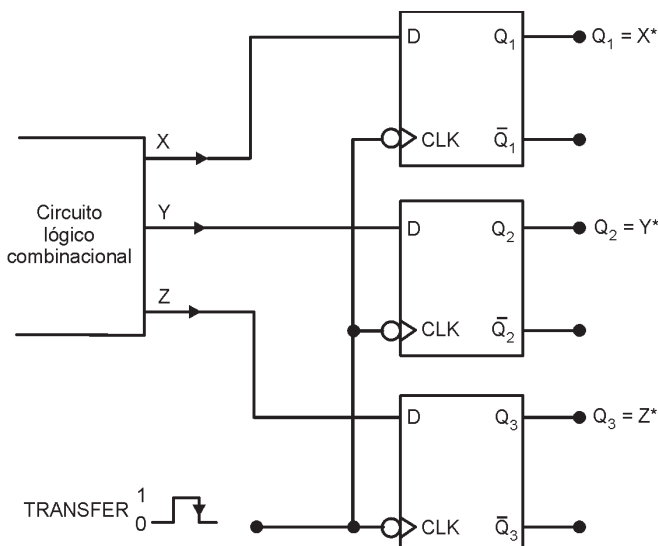
Um flip-flop tipo D pode ser implementado a partir de um flip-flop J-K, figura 17.



OBS. O mesmo procedimento pode ser usado para converter um flip-flop S-C em um flip-flop D.

Transferência de Dados Paralela

Na maioria dos aplicações do Flip-Flop D, a saída Q assume o valor da entrada D apenas em instantes precisamente definidos. Isso pode ser melhor visualizado na figura 18, a qual ilustra a transferência de dados paralela.



*Após a ocorrência da borda de descida

Figura 18

7.1.7. Latch D (Latch Transparente)

Como pode ser observado na figura 19, o latch D não é disparado por borda.

- Para $EN = 1$, D produzirá nível BAIXO em uma das entradas SET ou CLEAR $\Rightarrow Q$ terá mesmo nível lógico da entrada D. Se D mudar de nível enquanto EN for ALTO, a saída Q seguirá essas mudanças. Assim, $p/ EN = 1 \Rightarrow Q = D$.
- Qdo $EN = 0$, D estará desabilitada a alterar o Latch NAND, pois, as saídas das NANDS 1 e 2 \Rightarrow Nível Alto (Latch NAND repouso), Assim $p/ EN = 0 \Rightarrow Q$ não muda, mesmo se D mudar de estado.

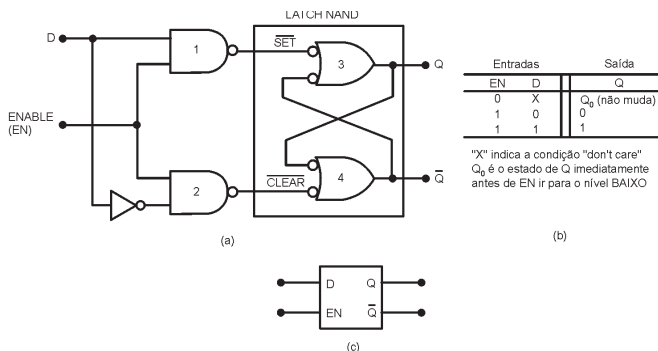


Figura 19

Exemplo 5: Determine a forma de onda na saída Q para um latch D com as formas de onda das entradas EN e D mostradas na figura abaixo. Considere inicialmente $Q=0$.

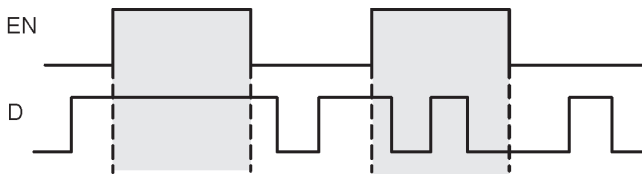
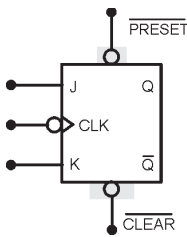


Figura 20 – Exemplo 5

7.1.8 Entradas Assíncronas

- Podem ser usadas para colocar o FF no estado 1 ou 0 em qualquer instante, independentemente das condições das outras entradas.
- São entradas de sobreposição, podendo ser usadas para sobrepor todas as outras entradas de forma a colocar o FF em um determinado estado.

A figura 21 mostra um flip-flop J-K COM clock e duas entradas assíncronas: $\overline{\text{PRESET}}$ e $\overline{\text{CLEAR}}$, as quais são ativadas em nível BAIXO.

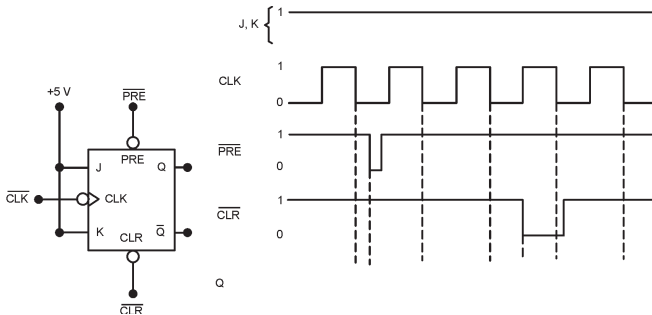


PRESET	CLEAR	Resposta do FF
1	1	Operação com clock*
0	1	Q = 1 (independente do CLK)
1	0	Q = 0 (independente do CLK)
0	0	Não usada

*Q responderá a JK e CLK

- $\overline{\text{PRESET}} = \overline{\text{CLEAR}} = 1$. Entradas assíncronas desativadas e FF responde às entradas J, K e CLK;
- $\overline{\text{PRESET}} = 0$ e $\overline{\text{CLEAR}} = 1$. é ativada e Q é imediatamente colocada em 1, quaisquer que sejam J, K e CLK;
- $\overline{\text{PRESET}} = 1$ e $\overline{\text{CLEAR}} = 0$. ativada e Q é imediatamente colocada em 0, quaisquer que sejam J, K e CLK;
- $\overline{\text{PRESET}} = \overline{\text{CLEAR}} = 0$. Não usada, resposta ambígua.

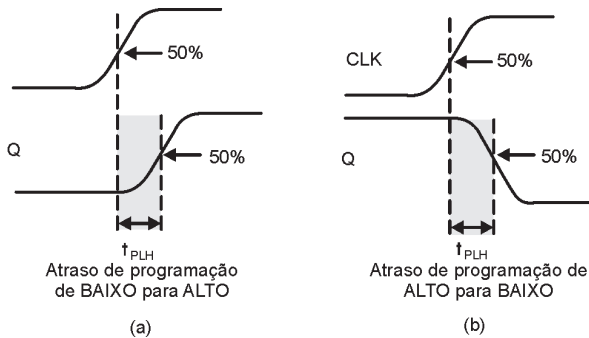
Exemplo 6: Determine a resposta de saída Q às formas de onda mostradas na figura 22. Considere Q inicialmente em nível ALTO.



7.1.9 Temporização em Flip-flops

- a) **Tempos de setup e hold:** os fabricantes especificam os valores mínimos de t_s e t_H para garantir o disparo confiável de flip-flops.

- b) **Atrasos de Propagação:** é o atraso de tempo a partir do instante em que o sinal é aplicado até o instante em que a saída comuta de estado. A figura 23 mostra os tempos:
- t_{PLH} = atraso para comutar do estado BAIXO para ALTO;
- t_{PHL} = atraso para comutar do estado ALTO para BAIXO.



- c) **Frequência Máxima de clock, $f_{MÁX}$:** é a maior frequência que pode ser aplicada na entrada CLK de um FF mantendo ainda um disparo confiável.
- d) **Tempos de duração do pulso de clock nos níveis ALTO e BAIXO** (figura 24(a)):
- $t_{W(L)}$: tempo mínimo que o CLK tem de permanecer no nível BAIXO antes de ir para o nível ALTO;
- $t_{W(H)}$: tempo mínimo que o CLK tem de permanecer no nível ALTO antes de ir para o nível BAIXO.
- e) **Largura de pulsos assíncronos ativos $t_{W(L)}$:** tempo mínimo de duração que a entrada CLEAR ou PRESET tem de permanecer no estado ativo de forma a setar ou resetar o FF de modo confiável, figura 24 (b)
- f) **Tempo de transição do clock:** é o tempo de subida e descida do sinal de clock, deve ser o menor possível para garantir o disparo confiável. Os fabricantes fornecem um parâmetro geral para todos os CIs de uma família lógica. Ex: tempo de transição ≤ 50 ns para dispositivos TTL e 200 ns para CMOS.

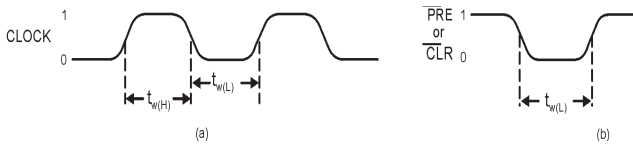
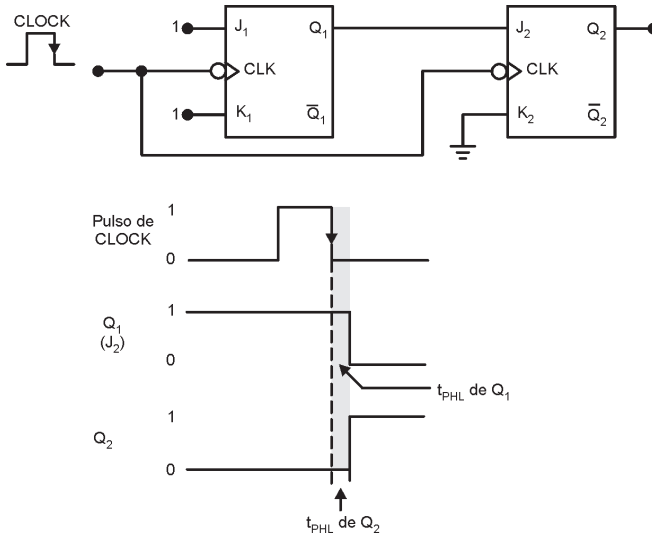


Figura 24 – (a) Tempos de duração do CLK em nível BAIXO e ALTO; (b) Largura do pulso assíncrono

Obs. A figura 25 ilustra um problema de temporização em circuitos com FFs:

- Como Q_1 muda de estado na borda de descida do pulso de clock, a entrada J_2 de Q_2 estará mudando de estado quando receber a mesma borda de descida do clock. Isso pode conduzir a uma resposta imprevisível de Q_2 .
- Q_2 responderá adequadamente ao nível lógico presente em Q_1 antes da borda de descida de CLK, desde que o t_H de Q_2 seja menor que o atraso de propagação de Q_1



Exemplo 7: Determine a saída Q p/ um flip-flop J-K disparado por borda negativa que tem como entrada as formas de onda mostradas abaixo. Considere $t_H = 0$ e que, inicialmente, $Q=0$.

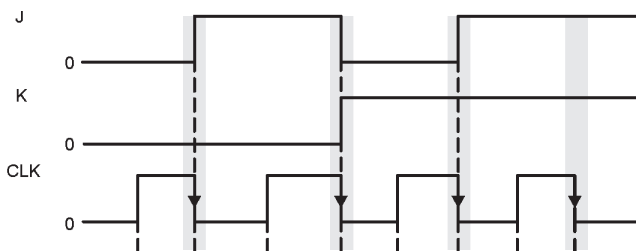


Figura 26 – Exemplo 7

7.1.10. Flip-Flop Mestre/Escravo

Este flip-flop proporciona uma maneira de se evitar a comutação para o estado oposto mais de uma vez durante uma borda positiva de relógio. Neste caso, o mestre é disparado pela borda positiva e o escravo pela borda negativa. Portanto, o mestre responde às entradas J e K antes do escravo

Por exemplo, se $J=1$ e $K=0$, o mestre é ativado na borda positiva do relógio. A saída Q alta do mestre aciona a entrada J do escravo. Assim, quando chega à borda negativa do relógio, o escravo é ativado, copiando a ação do mestre, figura 27.

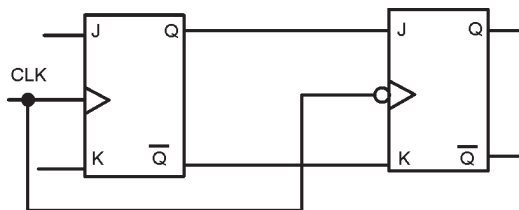


Figura 27 – Flip-flop J-K mestre/escravo

7.1.11 Flip-Flop do Tipo T

Este flip-flop é obtido a partir de um J-K mestre-escravo com as entradas J e K conectadas em curto. Obviamente, não irão ocorrer entradas do tipo $J=0 - K=1$ e $J=1 - K=0$.

A Fig. 28 mostra a ligação e o bloco representativo do flip-flop do tipo T, sensível à borda de descida dos pulsos de clock. A tabela verdade é mostrada a seguir, onde Q_A = saída anterior.

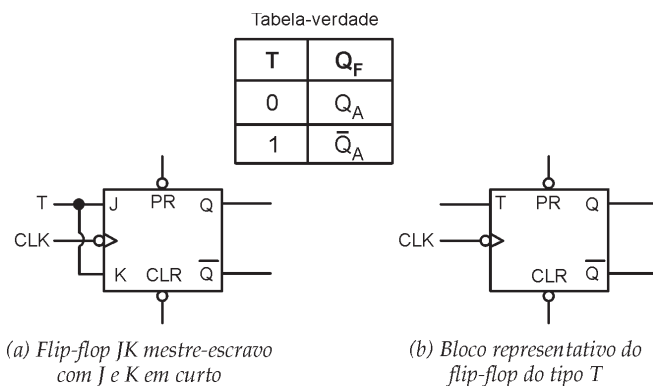


Figura 28 – Flip-flop Tipo T

7.2. Armazenamento e Transferência de Dados

O uso mais comum de Flip-Flops é no armazenamento de dados ou informações, que podem ser codificados em binário. Esses dados são geralmente armazenados em grupos de FFs denominados **Registadores**.

A operação mais comum realizada sobre os dados armazenados em FFs ou registradores é a operação de **Transferência de Dados**. Quando as entradas de controle síncronas e a entrada CLK são usadas realiza-se a **transferência síncrona**, conforme mostra a figura 29. Quando as entradas assíncronas são usadas para realizar a operação de transferência, realiza-se a **transferência assíncrona**, figura 30.

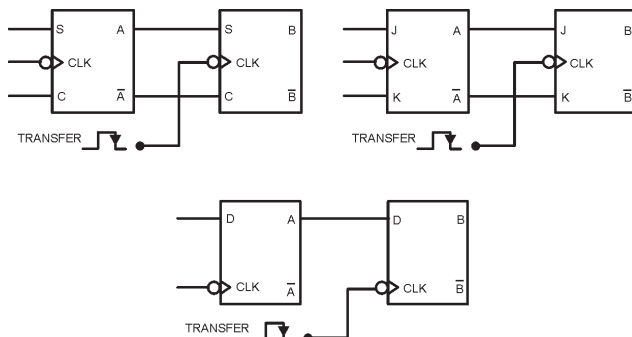


Figura 29 – Transferência Síncrona de Dados

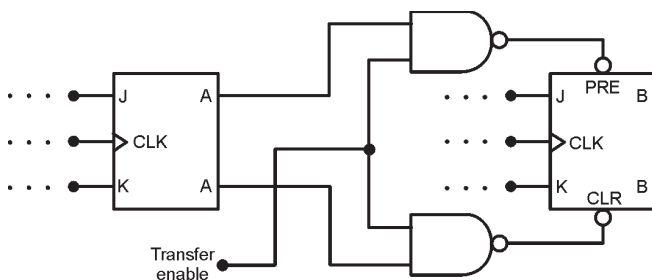
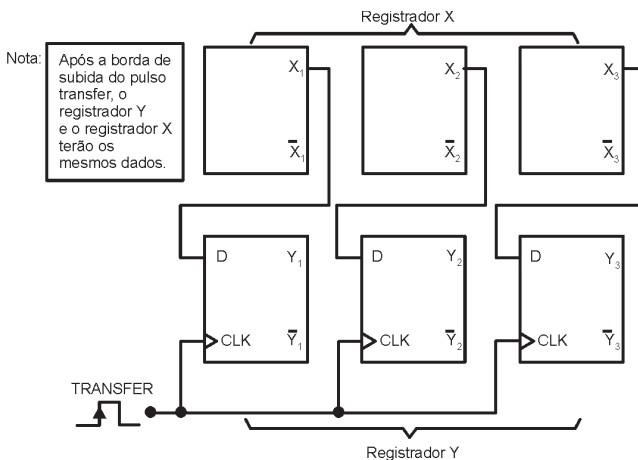


Figura 30 – Transferência Assíncrona de Dados

7.2.1 Transferência Paralela de Dados

A figura 31 mostra a transferência paralela de dados de um registrador para outro. A transferência paralela não altera o conteúdo do registrador que a fonte de dados. Por exemplo, se $X_1 X_2 X_3 = 101$ e $Y_1 Y_2 Y_3 = 011$ antes de ocorrer o pulso transfer, após ocorrer o pulso transfer o conteúdo dos dois registradores será 101.



7.2.2. Transferência Serial de Dados: Registradores de Deslocamento

Um **registrador** é simplesmente um grupo de flip-flops que pode ser usado para armazenar um número binário. Deverá haver um flip-flop para cada bit do número binário. Desta forma, para armazenar uma informação de mais de 1

bit, o sistema denominado de registrador de deslocamento pode ser utilizado.

A figura 32 mostra um registrador de deslocamento de 4 bits. Observe que quando ocorre uma borda de descida no pulso de deslocamento, cada FF recebe o valor armazenado previamente no FF à esquerda.

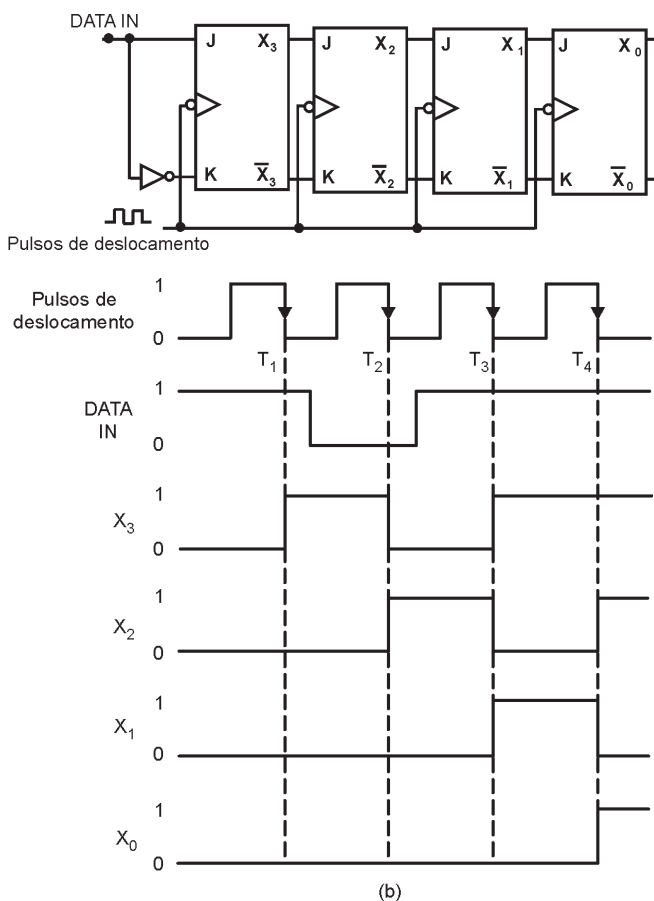


Figura 32 – Registrador de deslocamento de 4 bits

Obs. nesse tipo de registrador de deslocamento é necessário que os FFs tenham um tempo de hold (t_H) muito pequeno, porque existem momentos em que as entradas J e K estão mudando de estado no mesmo instante da transição do CLK.

A figura 33 mostra dois registradores de deslocamento de três bits conectados de modo que o conteúdo do registrador X seja transferido para o registrador Y, de forma serial. Pode-se observar que são utilizados flip-flops do tipo D, pois estes circuitos seqüenciais simplesmente registram na saída Q o valor do bit de entrada, seja ele 0 ou 1.

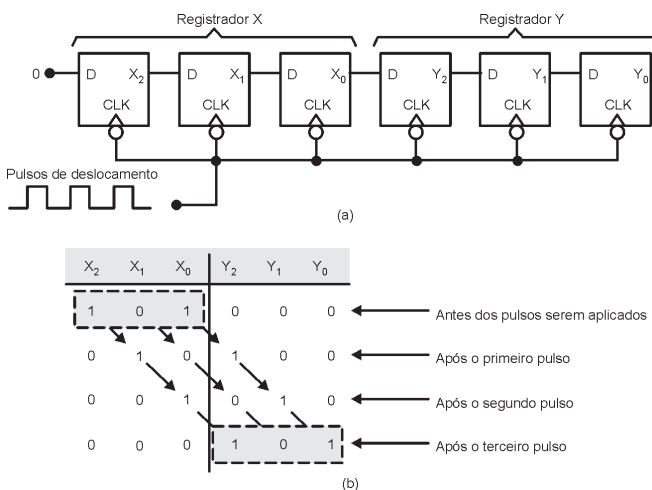


Figura 33 – Transferência serial de dados

7.3 Divisão de Frequência e Contagem

Na figura 34, cada FF tem suas entradas J e K em nível 1, para que ele mude de estado (comute) sempre que os sinais de sua entrada CLK for do nível ALTO para BAIXO.

Os pulsos de clock são aplicados apenas na entrada CLK do FF Q_0 . A saída Q_0 está conectada na entrada CLK do FF Q_1 , e a saída de Q_1 está conectada na entrada CLK do FF Q_2 .

- O FF Q_0 comuta na transição negativa de cada pulso na entrada de clock. Q_0 tem uma frequência que é exatamente a metade da frequência dos pulsos de clock;

- O FF Q1 comuta de estado cada vez que a saída Q0 vai do nível ALTO p/ BAIXO \Rightarrow a forma de onda de Q1 tem uma frequência que é exatamente a metade da frequência de Q0 \Rightarrow um quarto da frequência do sinal de clock;
- O FF Q2 comuta de estado cada vez que a saída Q1 vai do nível ALTO p/ BAIXO \Rightarrow a forma de onda de Q2 tem uma frequência que é exatamente a metade da frequência de Q1 \Rightarrow um oitavo da frequência do sinal de clock

OBS.: Se acrescentarmos um quarto FF (Q₃), ele teria uma frequência = 1/16 frequência de clock. Usando um número apropriado de flip-flops esse circuito pode dividir uma frequência por qualquer potência de 2: N flip-flops \Rightarrow frequência de saída do último FF = $1/2^N$ da frequência de entrada. Essa aplicação é conhecida como **divisor de frequência**.

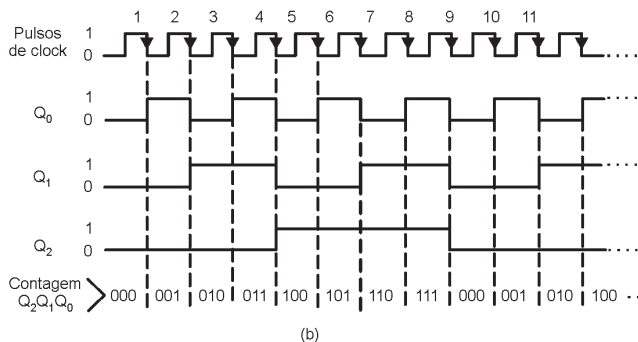
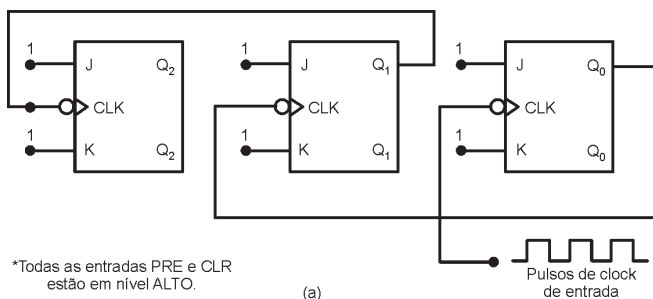


Figura 34 – Operação de divisão de frequência e contagem

O circuito da figura 34 também funciona como um **contador binário**. A tabela a seguir mostra a sequência de estados dos FFs após a ocorrência de cada pulso de clock para o circuito anterior. Observe que os estados resultantes correspondem ao número de pulsos ocorridos.

<u>2²</u>	<u>2¹</u>	<u>2⁰</u>	
Q ₂	Q ₁	Q ₀	
0	0	0	Antes de aplicar os pulsos de clock
0	0	1	Após o pulso #1
0	1	0	Após o pulso #2
0	1	1	Após o pulso #3
1	0	0	Após o pulso #4
1	0	1	Após o pulso #5
1	1	0	Após o pulso #6
1	1	1	Após o pulso #7
0	0	0	Após o pulso #8 retorna para 000
0	0	1	Após o pulso #9
0	1	0	Após o pulso #10
0	1	1	Após o pulso #11
.	.	.	.
.	.	.	.
.	.	.	.

Tabela 1

Outra forma de representar a mudança de estados dos FFs com os pulsos aplicados é através do diagrama de transição de estados mostrado na figura 35, no qual cada círculo representa um estado possível.

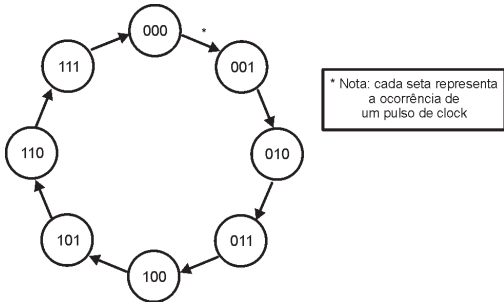


Figura 35 – Diagrama de transição de estados

Módulo do Contador:

Indica o número de estados da sequência de contagem. Para N flip-flops o contador resultante terá 2^N estados diferentes, e portanto, será um contador de módulo 2^N .

O valor do módulo de um contador indica também a razão entre a frequência de entrada e a frequência obtida na saída do último flip-flop. Por ex: um contador de 4 bits possui quatro FFs, sendo um contador de módulo $2^4 = 16$. Portanto esse contador pode contar até 15 ($2^4 - 1$), ele também pode ser usado p/ dividir a frequência por 16.

Exemplo 8: Considere um circuito de um contador que possui seis FFs conectados segundo o diagrama da figura 34 (isto é, $Q_5, Q_4, Q_3, Q_2, Q_1, Q_0$).

- Determine o módulo do contador.
- Determine a frequência na saída do último FF (Q_5) quando a frequência do clock de entrada for de 1 MHz.
- Qual a faixa de estados de contagem desse contador?
- Considere como estado (contagem) inicial o valor 000000. Qual será o estado do contador após 129 pulsos.

7.3.1. Contadores de Módulo $< 2^N$

Um contador básico pode ser modificado para gerar um módulo $< 2^N$ fazendo com que o contador pule estados que normalmente fazem parte da sequência de contagem. A figura 36 mostra um contador de módulo de módulo 6.

As entradas da porta NAND são as saídas dos FFs B e C. P a saída da porta NAND irá p/ o nível BAIXO sempre que $B = C = 1$. Essa condição ocorre quando o contador passa do estado 101 para 110 na transição negativa do pulso 6. O nível BAIXO na saída da porta NAND resetará imediatamente (geralmente em poucos ns) o contador para o estado 000. A saída da porta NAND retorna para 1 visto que a condição $B=C=1$ não existe mais.

Qualquer valor de módulo desejado pode ser obtido alterando-se as entradas da porta NAND.

Procedimento Geral para Construção de Contadores:

1. Determine o menor número de FFs de forma que $2^N \geq X$ e conecte-os como um contador. Se $2^N = X$, dispense os passo 2 e 3.
2. Conecte a saída de uma porta NAND às entradas assíncronas CLEAR de todos os FFs;
3. Determine quais são os FFs que estarão em nível ALTO na contagem = X; então conecte as saídas normais desses FFs às entradas da porta NAND.

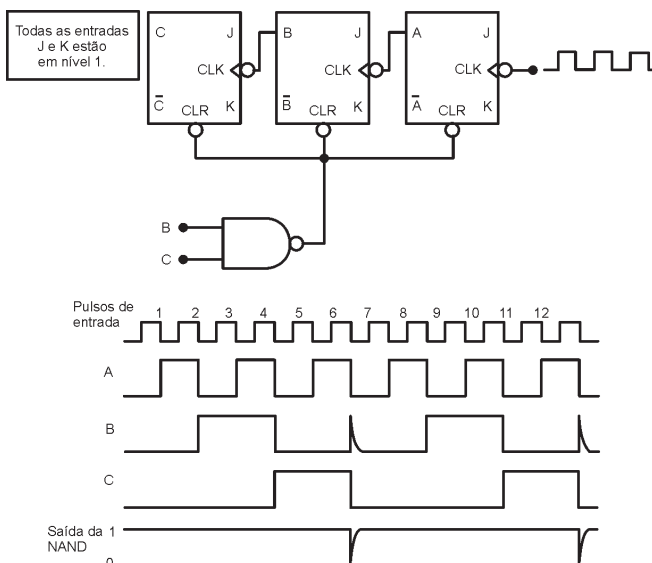


Figura 36 – Contador de Módulo 6

7.3.2 Contador Decrescente

Os contadores estudados realizavam a contagem crescente dos bits. Propõe-se, agora, estudar um circuito lógico que efetua a contagem decrescente.

O circuito que efetua a contagem decrescente é o mesmo que realiza a contagem crescente, com a diferença de se extrair as saídas dos terminais barrados. A Fig. 37 apresenta o circuito do contador assíncrono decrescente de módulo 8

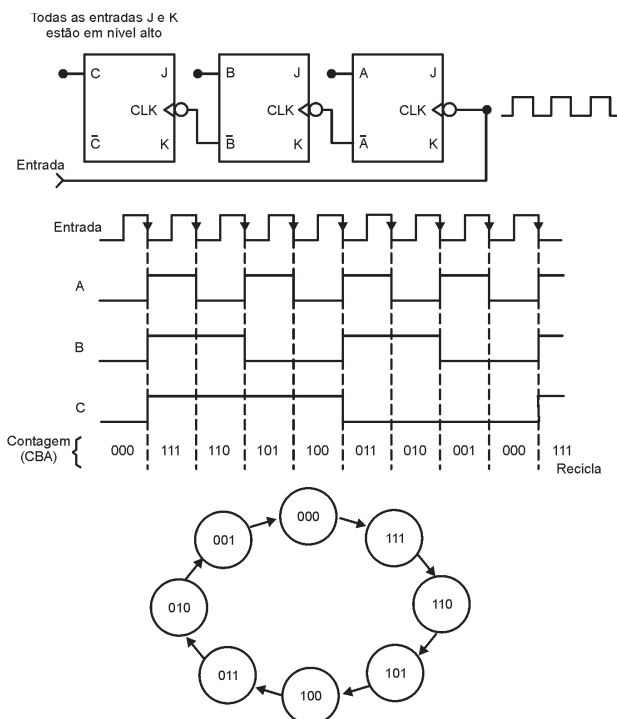


Figura 37 – Contador Decrescente de Módulo 8

Nos contadores apresentados anteriormente a saída de cada FF aciona a entrada CLK do FF seguinte, por isso são denominados **contadores assíncronos** porque os FFs não mudam de estado exatamente com o mesmo sincronismo com que os pulsos são aplicados. O inconveniente desse tipo de contador é justamente o acúmulo dos atrasos de propagação. Porém, essas limitações podem ser superadas com os **contadores síncronos ou paralelos**, nos quais os FFs são disparados simultaneamente pelos pulsos do clock de entrada.

EXERCÍCIOS PROPOSTOS:

Resolva os problemas 5.1 a 5.3, 5.7 a 5.22, 5.27 a 5.36 e 5.44 do capítulo 5 do **Livro Texto**.

Resolva os problemas 6.21, 6.22, 6.24, 6.26 e 6.27 do capítulo 5 do **Livro Texto**.

Resolva os problemas 7.1 a 7.4, 7.8, 7.14 a 7.17 do capítulo 7 do **Livro Texto**.

CONSULTAS RECOMENDADAS:

Livro Texto: Sistemas Digitais: Princípios e Aplicações, 8ª
Edição - Ronald J. Tocci e Neal S. Widmer – São Paulo:
Prentice Hall, 2003 – Capítulos: 5, 6 e 7.

